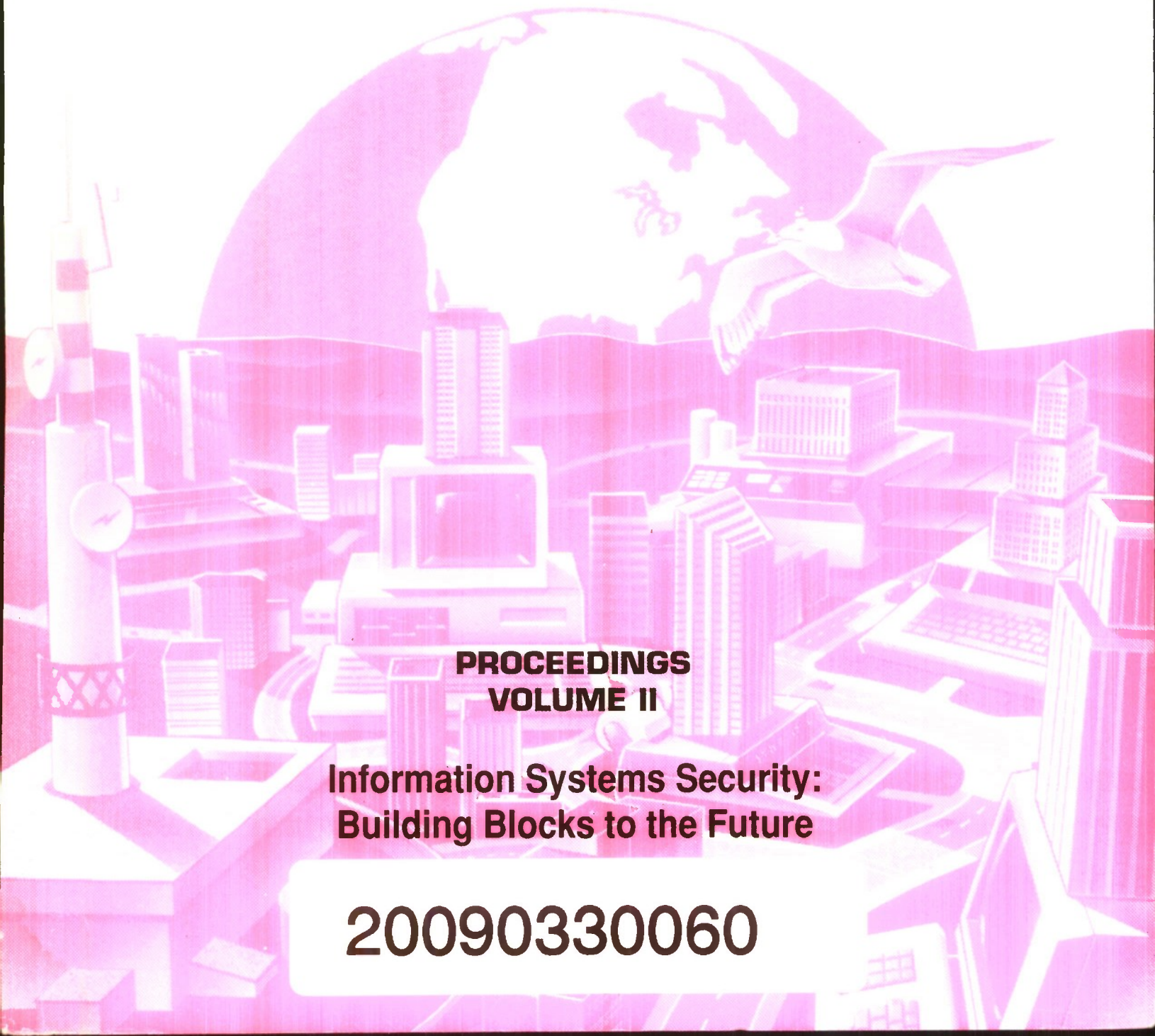
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY/
NATIONAL COMPUTER SECURITY CENTER

15TH NATIONAL COMPUTER SECURITY CONFERENCE

October 13-16, 1992
Baltimore Convention Center
Baltimore, MD

APR 15



PROCEEDINGS
VOLUME II

Information Systems Security: Building Blocks to the Future

20090330060

DEFENSE TECHNICAL INFORMATION CENTER

Information for the Defense Community

DTIC® has determined on 04/10/2009 that this Technical Document has the Distribution Statement checked below. The current distribution for this document can be found in the DTIC® Technical Report Database.

☒ **DISTRIBUTION STATEMENT A.** Approved for public release; distribution is unlimited.

☐ **© COPYRIGHTED**; U.S. Government or Federal Rights License. All other rights and uses except those permitted by copyright law are reserved by the copyright owner.

☐ **DISTRIBUTION STATEMENT B.** Distribution authorized to U.S. Government agencies only (fill in reason) (date of determination). Other requests for this document shall be referred to (insert controlling DoD office)

☐ **DISTRIBUTION STATEMENT C.** Distribution authorized to U.S. Government Agencies and their contractors (fill in reason) (date of determination). Other requests for this document shall be referred to (insert controlling DoD office)

☐ **DISTRIBUTION STATEMENT D.** Distribution authorized to the Department of Defense and U.S. DoD contractors only (fill in reason) (date of determination). Other requests shall be referred to (insert controlling DoD office).

☐ **DISTRIBUTION STATEMENT E.** Distribution authorized to DoD Components only (fill in reason) (date of determination). Other requests shall be referred to (insert controlling DoD office).

☐ **DISTRIBUTION STATEMENT F.** Further dissemination only as directed by (inserting controlling DoD office) (date of determination) or higher DoD authority.

Distribution Statement F is also used when a document does not contain a distribution statement and no distribution statement can be determined.

☐ **DISTRIBUTION STATEMENT X.** Distribution authorized to U.S. Government Agencies and private individuals or enterprises eligible to obtain export-controlled technical data in accordance with DoDD 5230.25; (date of determination). DoD Controlling Office is (insert controlling DoD office).

Welcome!

The National Computer Security Center (NCSC) and the Computer Systems Laboratory (CSL) are pleased to welcome you to the Fifteenth Annual National Computer Security Conference. We believe that the Conference will stimulate a vital and dynamic exchange of information and foster an understanding of emerging technologies.

*The theme for this year's conference, "**Information Systems Security: Building Blocks to the Future**," reflects the continuing importance of the broader information systems security issues facing us. At the heart of these issues are two items which will receive special emphasis this week--Information Systems Security Criteria (and how it affects us), and the actions associated with organizational accreditation. These areas will be highlighted by emphasizing how organizations are integrating information security solutions. You will observe how Government, Industry, and Academe are cooperating to extend the state-of-the-art technology to information systems security. Presentations will provide you with some thoughtful insights as well as innovative ideas in developing your own solutions. Additionally, panel members will address how they develop their automated information security responsibilities. This cooperative educational program will refresh us with the perspectives of the past, and will project directions of the future.*

We firmly believe that awareness and responsibility are the foundations of all information security programs. For our collective success, we ask that you reflect on the ideas and information presented this week; then share this information with your peers, your management, your administration, and your customers. By sharing this information, we will develop a stronger knowledge base for tomorrow's journey.

PATRICK R. GALLAGHER, JR
Director
National Computer Security Center

JAMES H. BURROWS
Director
Computer Systems Laboratory

Conference

Dr. Marshall Abrams	*The MITRE Corporation*
Roland Albert	*Department of Defense*
James P. Anderson	*J.P.Anderson Company*
Devolyn Arnold	*Department of Defense*
James Arnold	*Department of Defense*
V.A. Ashby	*The MITRE Corporation*
David Balenson	*Trusted Information Systems, Inc.*
Dr. D. Elliott Bell	*BBND*
James W. Birch	*Secure Systems, Inc.*
W.Earl Boebert	*Secure Computing Technology Corporation*
Edward Borodkin	*National Computer Security Center*
Dr. Martha Branstad	*Trusted Information Systems, Inc.*
Dr.Blaine Burnham	*Department of Defense*
Dr. John Campbell	*Department of Defense*
David Chizmadia	*Department of Defense*
Dr. Deborah Cooper	*Unisys*
Donna Dodson	*National Institute of Standards and Technology*
Dr. Deborah Downs	*The AEROSPACE Corporation*
David Ferraiolo	*National Institute of Standards and Technology*
Ellen Flahavin	*National Institute of Standards and Technology*
L. Dain Gary	*Carnegie Mellon University*
William Geer	*AFCSC*
Virgil Gibson	*Grumann Data Systems*
Dennis Gilbert	*National Institute of Standards and Technology*
Irene Gilbert	*National Institute of Standards and Technology*
Captain James Goldston, USAF	*AFCSC*
Dr. Joshua Guttman	*The MITRE Corporation*
Dr. Grace Hammonds	*AGCS, Inc.*
Douglas Hardie	*Unisys Corporation*
Ronda Henning	*Harris Corporation*
Dr. Harold Highland, FICS	*Compulit, Inc.*
Jack Holleran	*National Computer Security Center*
Hilary H. Hosmer	*Data Security, Inc.*
Russell Housley	*XEROX Information Systems*
Howard Israel	*AT&T Bell Laboratories*

Referees

Professor Sushil Jajodia	*George Mason University*
John Keenan	*CISEC*
Dr. Richard Kemmerer	*University Of California, Santa Barbara*
Dr. Steven Kent	*BBN*
Richard Kuhn	*National Institute of Standards and Technology*
Steven LaFountain	*Department of Defense*
Paul A. Lambert	*Motorola GEG*
Dr. Carl Landwehr	*Naval Research Laboratory*
Robert Lau	*Department of Defense*
Dr. Theodore M.P. Lee	*Trusted Information Systems, Inc.*
Steven B. Lipner	*The MITRE Corporation*
Teresa Lunt	*SRI International*
Frank Mayer	*Aerospace Corporation*
Dr. Catherine Meadows	*Naval Research Laboratory*
Sally Meglathery	*New York Stock Exchange*
William H. Murray	*Deloitte & Touche*
Noel Nazario	*National Institute of Standards and Technology*
Dr. Peter Neumann	*SRI International*
Nick Pantiuk	*Grumann Data Systems*
Donn Parker	*SRI International*
Dr. Charles Pfleeger	*Institute for Defense Analyses*
Professor Ravi Sandhu	*George Mason University*
Marvin Schaefer	*CTA, Inc.*
Daniel Schnackenberg	*Boeing Aerospace Corporation*
Miles Smid	*National Institute of Standards and Technology*
Brian Snow	*Department of Defense*
Dr. Dennis Steinauer	*National Institute of Standards and Technology*
Mario Tinto	*Department of Defense*
Eugene Troy	*National Institute of Standards and Technology*
Kenneth vanWyk	*Carnegie Mellon University*
Grant Wagner	*Department of Defense*
Major Glenn Watt, USAF	*USAF Strategic Air Command*
Wayne Weingaertner	*Department of Defense*
Howard Weiss	*SPARTA*
Roy Wood	*Department of Defense*

Awards Ceremony

6:00 p.m., Thursday, October 15
Convention Center, Terrace Level

A joint awards ceremony will be held at which the National Institute of Standards and Technology (NIST) and the National Computer Security Center (NCSC) will honor the vendors who have successfully developed products meeting the standards of the respective organizations.

The Computer Security Division at NIST provides validation services for vendors to use in testing devices for conformance to security standards defined in three Federal Information Processing Standards (FIPS): FIPS 46-1, *The Data Encryption Standard (DES)*; FIPS 113, *Computer Data Authentication*; and FIPS 171, *Key Management Using ANSI X9.17.*

Conformance to FIPS 46-1 is tested using the Monte Carlo test described in NBS Special Publication 500-20, *Validating the Correctness of Hardware Implementations of the NBS Data Encryption Standard* which requires performing eight million encryptions and four million decryptions.

Conformance to FIPS 113 and its American Standards Institute counterpart, ANSI X9.9, *Financial Institution Message Authentication (Wholesale)* is tested using an electronic bulletin board (EBB) test as specified in NBS Special Publication 500-156, *Message Authentication Code (MAC) Validation System: Requirements and Procedures.* The test consists of a series of challenges and responses in which the vendor is requested to either compute or verify an MAC using a specified randomly generated key.

Conformance to FIPS 171, which adopts ANSI X9.17, *Financial Institution Key Management (Wholesale)*, is also tested using an EBB as specified in a document entitled *NIST Key Management Validation System Point-to-Point (PTP) Requirements.*

The NCSC recognizes vendors who contribute to the availability of trusted products and thus expand the range of solutions from which customers may select to secure their data. The products are placed on the Evaluated Products List (EPL) following a successful evaluation against the *Trusted Computer Systems Evaluation Criteria* including its interpretations: *Trusted Database Interpretation*, *Trusted Network Interpretation*, and *Trusted Subsystem Interpretation*. Vendors who have completed the evaluation process will receive a formal certificate of completion from the Director, NCSC marking the addition to the EPL. In addition, vendors will receive honorable mention for being in the final stages of an evaluation as evidenced by transition into the Formal Evaluation phase or for placing a new release of a trusted product on the EPL by participation in the Ratings Maintenance Program. The success of the Trusted Product Evaluation Program is made possible by the commitment of the vendor community.

We congratulate all who have earned these awards.

15th National Computer Security Conference
Table of Contents
Refereed Papers

1 Accreditation: Is it a Security Requirement or a Good Management Practice?
Thomas E. Anderson, USATREX International Inc.

9 Application Layer Security Requirements of a Medical Information System
Deborah Hamilton, Hewlett-Packard Laboratories

18 An Approach for Multilevel Security (MLS) Acquisition
Bill Neugent, The MITRE Corporation

28 Architectural Implications of Covert Channels
Norman E. Proctor, Peter G. Neumann,
Computer Science Lab, SRI International

44 Assessing Modularity in Trusted Computing Bases
J. L. Arnold, R. J. Bottomly, National Security Agency
D. B. Baker, D. D. Downs, The Aerospace Corporation
F. Belvin, S. Chokhani, The MITRE Corporation

57 Companion Document Series to the Trusted Database Management System Interpretation
LouAnna Notargiacomo, Victoria Ashby, Vinti Doshi, Jarellann Filsinger, Sushil Jajodia, The MITRE Corporation
Lieutenant Colonel Ron Ross, USA, National Computer Security Center

66 Computer Security and Total Quality Management
Major Gregory B. White, USAF Academy
Mr. Lee Sutterfield, AFCSC/SRO
Mr. Chuck Arvin, CTA

76 Concept for a Smart Card Kerberos
Marjan Krajewski, Jr., The MITRE Corporation

84 Concept Paper--An Overview of the Proposed Trust Technology Assessment Program
Ellen E. Flahavin, Patricia R. Toth, Computer Security Division, National Institute of Standards and Technology

93 Current Endorsed Tools List (ETL) Examples Research Lessons Learned
Cristi Garvey, Aaron Goldstein, Eric Anderson,
TRW Systems Integration Group

101 Data Security for Personal Computers
Paul Bicknell, The MITRE Corporation

111 Defense Against Computer Aids
Horace B. Peele, Air Force Intelligence Command

120 E-Mail Privacy and the Law
Christine Axsmith, Esq., ManTech Strategic Associates

126 Electronic Measurement of Software Sharing for Computer Virus Epidemiology
Larry de La Beaujardiere, Department of Computer Science, University of California

134 Enforcing Entity and Referential Integrity in Multilevel Secure Databases
Vinti M. Doshi, Sushil Jajodia, The MITRE Corporation

144 Evolving Criteria for Evaluation: The Challenge for the International Integrator of the 90s
Virgil Gibson, Joan Fowler, Grumman Data Systems

153 An Example Complex Application for High-Assurance Systems
Frank L. Mayer, The Aerospace Corporation
Steven J. Padilla, SPARTA, Inc.

165 Experience with a Penetration Analysis Method and Tool
Sarbari Gupta, Virgil D. Gligor,
Electrical Engineering Department, University of Maryland

184 Extending Our Hardware Base: A Worked Example
Noelle McAuliffe, Trusted Information Systems, Inc.

194 Finding Security Flaws in Concurrent and Sequential Designs Using Planning Techniques
Deborah A. Frincke, Myla Archer, Karl Levitt,
Division of Computer Science, University of California, Davis

204 A Foundation for Covert Channel Analysis
Todd Fine, Secure Computing Corporation

213 General Issues to be Resolved in Achieving Multilevel Security (MLS)
Bill Neugent, The MITRE Corporation

221 Implementation Considerations for the Typed Access Matrix Model in a Distributed Environment
Ravi S. Sandhu, Gurprett S. Suri, Center for Secure Information Systems & Department of Information and Software Systems Engineering, George Mason University

236 Implications of Monoinstantiation in a Normally Polyinstantiated Multilevel Secure Database
Frank E. Kramer, Steven M. Heffern, Digital Equipment Corporation

244 Information System Security Engineering: Cornerstone to the Future
Dr. Donald M. Howe, National Security Agency

262 Internetwork Security Monitor: An Intrusion-Detection System for Large-Scale Networks
L. T. Heberlein, B Mukherjee, K. N. Levitt, Computer Security Laboratory, Division of Computer Science, University of California

252 Integrity and Assurance of Service Protection in a Large, Multipurpose, Critical System
Howard L. Johnson, Information Intelligence Sciences, Inc.
Chuck Arvin, Earl Jenkinson, CTA Incorporated
Captain Bob Pierce, AF Cryptologic Support Center, Hq. AFIC, AFCSC/SR

272 Intrusion and Anomaly Detection: ISOA Update
J. R. Winkler, J. C. Landry, PRC, Inc.

282 Issues in the Specification of Secure Composite Systems
Judith Hemenway, Dan Gambel, Grumman Data Systems

292 Issues to Consider when using Evaluated Products to Implement Secure Mission Systems
Lieutenant Colonel William R. Price, USAF, Headquarters Air Force Space Command (LKXS)

300 The IT Security Evaluation Manual (ITSEM)
Y. Klein, Service Central de la Sécurité des Systèmes d'Information, Paris, France
E. Roche, Department of Trade and Industry, London, United Kingdom
F. Taal, Netherlands National Communications Security Agency, The Hague, The Netherlands
M. Van Dulm, Ministry of the Interior, The Hague, The Netherlands
U. Van Essen, German Information Security Agency, Bonn, Germany
P. Wolf, Centre D'Électronique de l'Armement, Bruz, France
J. Yates, Communications-Electronics Security Group, Cheltenham, United Kingdom

310 The Kinetic Protection Device
Gregory Mayhew, Richard Frazee, Mark Bianco, Hughes Aircraft Company Ground Systems Group

319 Knowledge-Based Inference Control in a Multilevel Secure Database Management System
Bhavani Thuraisingham, The MITRE Corporation

329 A Lattice Interpretation of the Chinese Wall Policy
Ravi S. Sandhu, Center for Secure Information Systems & Department of Information and Software Systems Engineering, George Mason University

340 A Local Area Network Security Architecture
Lisa J. Carnahan, National Institute of Standards and Technology

350 Mandatory Policy Issues of High Assurance Composite Systems
Jonathan Fellows, Grumman Data Systems

359 Mediation and Separation in Contemporary Information Technology Systems
Marshall D. Abrams, Jody E. Heaney, Michael V. Joyce, The MITRE Corporation

369 Metapolicies II
Hilary H. Hosmer, Data Security Inc.

379 A Model for the Measurement of Computer Security Posture
Lee Sutterfield, Todd Schell, Gregory White, Kent Doster, Don Cuiskelly, United States Air Force

389 A Model of Risk Management in the Development Life Cycle
Captain Charles R. Pierce, USAF, Air Force Cryptologic Support Center

399 A Multilevel Secure Database Management System Benchmark
Linda M. Schlipper, Jarrellann Filsinger, Vinti M. Doshi, The MITRE Corporation

409 The Multipolicy Paradigm
Hilary H. Hosmer, Data Security Inc.

423 The Need for a Multilevel Secure (MLS) Trusted User Interface
Greg Factor, Steve Heffern, Doug Nelson, Jim Studt, Mary Yelton, Digital Equipment Corporation

429 Network Security Via DNSIX, Integration of DNSIX and CMW Technology
Howard A. Heller, Harris Corporation

438 New Dimensions in Data Security
Karl Heinz Mundt, CE Infosys

448 A Note on Compartmented Mode: To B2 or not B2?
Theodore M. P. Lee, Trusted Information Systems, Inc.

459 Operating System Support for Trusted Applications
Richard Graubart, The MITRE Corporation

467 Operational Support of Downgrading in a Multi-Level Secure System
Doug Nelson, Greg Factor, Jim Studt, Mary Yelton, Steve Heffern, Frank Kramer, Digital Equipment Corporation

473 PM: a Unified Automated Deduction Tool for Verification
George Fink, Lie Yang, Myla Archer, University of California, Davis

482 Potential Benefits from Implementing the Clark-Wilson Integrity Model Using an Object-Oriented Approach
Craig A. Schiller, Science Applications International Corporation

494 Precise Identification of Computer Viruses
Lawrence E. Bassham III, W. Timothy Polk,
National Institute of Standards and Technology

503 Priorities for LAN Security - A Case Study of a Federal Agency's LAN Security
Shu-jen H. Chang, National Institute of Standards and Technology

513 Protected Groups: An Approach to Integrity and Secrecy in an Object-Oriented Database
James M. Slack, Computer and Information Sciences Department,
Mankato State University
Elizabeth A. Unger, Department of Computing and Information Sciences,
Kansas State University

523 Provably Weak Cryptographic Systems
John Higgins, Brigham Young University, Computer Science Department
Cameron Mashayeki, WordPerfect Corporation

534 Re-Use of Evaluation Results
Jonathan D. Smith, Admiral Management Services Ltd. Commercial Licensed Evaluation Facility, U.K.

544 Risk Management of Complex Networks
Richard Cox, Dr. Michael O'Neill, CTA Incorporated
Lieutenant Colonel William Price, HQ AFSPACECOM/LKXS

554 Role-Based Access Controls
David Ferraiolo, Richard Kuhn,
National Institute of Standards and Technology

564 An SDNS Platform for Trusted Products
Ernie Borgoyne, Motorola Inc.
Ralph G. Puga, Trusted Information Systems, Inc.

574 SDNS Security Management
Wayne A. Jansen, National Institute of Standards and Technology

584 Security Management: Using the Quality Approach
Richard W. Owen, Jr., Computer Security Official Mission Operations Directorate, Johnson Space Center, NASA

593 A Security Reference Model for a Distributed Object System and its Application
Vijay Varadharajan, Hewlett-Packard Labs

620 Security Within the DODIIS Reference Model
Brian W. McKenney, The MITRE Corporation

631 Separation Machines
Jon Graff, Amdahl Corporation

641 Software Forensics: Can We Track Code to its Authors?
Eugene H. Spafford, Department of Computer Sciences, Purdue University
Stephen A. Weeber, Lawrence Livermore National Laboratory

651 Some More Thoughts on the Buzzword "Security Policy"
David M. Chizmadia, National Security Agency

661 Standard Certification - Progression
Captain Charles R. Pierce, USAF, Air Force Cryptologic Support Center

670 A Tamper-Resistant Seal for Trusted Distribution and Life-Cycle Integrity Assurance
Mark Bianco, Hughes Aircraft Company

680 A TCB Subset for Integrity and Role-Based Access Control
Daniel F. Sterne, Trusted Information Systems, Inc.

697 A Tool for Covert Storage Channel Analysis of the UNIX Kernel
David A. Willcox, Steve R. Bunch, Motorola Microcomputer Group

707 Toward a Model of Security for a Network of Computers
William H. Murray, Deloitte & Touche
Patrick Farrell, Department of Computer Science, George Mason University

717 Towards a Policy-Free Protocol Supporting a Secure X Window System
Mark Smith, AT&T Bell Laboratories

728 Use of a CASE Tool to Define the Specifications of a Trusted Guard
Robert Lazar, The MITRE Corporation
James H. Gray, III, Computer Sciences Corporation

Tutorials [Track D, Room 301-303]

738 Tutorial Series on Trusted Systems
R. Kenneth Bauer, Joel Sachs, Dr. Gary Smith,
Dr. William Wilson,
Arca Systems, Inc.
Dr. Charles Abzug, LtCdr Alan Liddle, Royal Navy,
Howard Looney,
Information Resources Management College, National Defense University

EXECUTIVE SUMMARIES

740 **Panel:** Addressing U. S. Government Security Requirements for OSI
Noel A. Nazario, Chair, National Institute of Standards and Technology
Ted Humphreys, XISEC Consultants Ltd., U.K.
Thomas C. Bartee, Institute for Defense Analysis
Dale Walters, Systems and Networks Architecture Division, National Institute of Standards and Technology

744 *Point of view:* OSE Implementor's Agreements
Dale Walters, National Institute of Standards and Technology

746 *Point of view:* Emerging OSI Security Protocols & Techniques
Ted Humphreys, XISEC Consultants Ltd., England

752 *Point of view:* Security Labels in OSI
T. C. Bartee, Institute for Defense Analyses

754 **Panel:** Challenges Facing Certification and Accreditation Efforts of the Military Services
Lieutenant Colonel Ron Ross, Chair, USA
Larry Merritt, AFCSC
Robert Zomback, CECOM
John Mildner, NESSEC

758 **Panel:** Domestic Privacy: Roll of Honor and Hall of Shame
Wayne Madsen, Chair

761 **Panel:** Health Issues Program
Gerald S. Long, Chair, Harrison Avenue Corporation

762 *Point of View:* The Benefits of Smart Card Technology in the Health Industry
Peter M. Fallon, Toshiba American Information Systems

764 *Point of View:* National Health Card
B. Bahramian, Beta Management Systems, Inc.

765 *Point of View:* The Optical Card as a Portable Medical Record
Stephen D. Price-Francis, Canon-Canada, Inc.

766 *Point of View:* Patient Data Confidentiality in the Health Care Environment
Marc Schwartz, Summit Medical Services, Inc.

768 **Panel:** Information Technology Security Requirements
D. Gilbert, Chair, National Institute of Standards and Technology
N. Lynch, National Institute of Standards and Technology
Dr. W. Maconochy, National Security Agency
S. Pitcher, Department Of Commerce
M. Swanson, National Institute of Standards and Technology

770 **Panel:** International Data Privacy: Roll of Honor and Hall of Shame
Wayne Madsen, Chair

774 **Panel:** Multilevel Security (MLS) Prototyping and Integration: Lessons Learned and DoD Directions
C. West, Chair, Defense Information Systems

775 **Workshop:** New Security Paradigm Workshop
Hilary Hosmer, Chair, Data Security, Inc.

777 *Point of view:* Managing Complexity in Secure Networks
Dr. David Bailey, Galaxy Computer Services

784 *Point of view:* A New Paradigm for Trusted Systems
Dr. Dorothy E. Denning, Georgetown University

792 **Panel:** Perspectives and Progress on International Criteria
Eugene Troy, Chair, National Institute of Standards and Technology
Lieutenant Colonel Ron Ross, USA
D. Ferraiolo, National Institute of Standards and Technology
Eugene Bacic, Canadian System Security Centre
Jonathan Wood, Department of Trade and Industry, U.K.

795 **Panel:** Perspectives on MLS System Solution Acquisition - A Debate by the Critical Players Involved
Joel E. Sachs, Chair, Arca Systems, Inc.

799 **Panel:** Security Protocols for Open Systems
Paul A. Lambert, Motorola, Inc.
David Solo, BBN
Doug Maughan, National Secuirty Agency
Russell Housley, Xerox
Dale Walters, National Institute of Standards and Technology
Mike White, Booz Allen & Hamilton

800 **Panel:** "TMach" A Symbol of International Harmonization
Ellen E. Flahavin, Chair, NIST
Brian Boesch, DARPA
Dr. Martha Branstad, Trusted Information Systems, Inc.
C. Ketley, U.K. Government
Klaus Keus, German Government

801 **Panel:** The Trusted Product Evaluations Program Process Action Team
S. Nardone, Chair, National Security Agency

802 **Panel:** Virus Attacks and Counterattacks Real-World Experiences
James P. Litchko, Chair, Trusted Information Systems, Inc.
Janet Keys, Headquarters NASA
Louise Mandeville, Miller, Balis & O'Neil, P.C.
George Wellham, MNC Financial, Inc.

Abrams, M. D. 359
Abzug, C. 738
Archer, M. 473
Ashby, V. 57
Anderson, E. 93
Anderson, T. E. 1
Archer, M. 194
Arnold, J. L. 44
Arvin, C. 66, 252
Axsmith, C., Esq. 120
Bacic, E. 792
Bahramian, B. 764
Bailey, D. 777
Baker, D. B. 44
Bartee, T. C. 752
Bassham III, L. E. 494
Bauer, R. K. 738
Belvin, F. 44
Bianco, M. 310, 670
Bicknell, P. 101
Boesch, B. 800
Borgoyne, E. 564
Bottomly, R.J. 44
Branstad, M. 800
Bunch, S. R. 697
Carnahan, L. J. 340
Chang, S. H. 503
Chizmadia, D. M. 651
Chokhani, S. 44
Cox, R. 544
Cuiskelly, D. 379
de La Beaujardiere, L. 126
Denning, D. E. 784
Doshi, V. 57, 134, 399
Doster, K. 379
Downs, D. D. 44
Factor, G. 423, 467
Fallon, P. M. 762
Farrell, P. 707
Fellows, J. 350
Ferraiolo, D. 554, 792
Filsinger, J. 57, 399
Fine, T. 204
Fink, G. 473
Flahavin, E. E. 84, 800
Fowler, J. 144
Frazee, R. 310
Frinck, D. A. 194
Gambel, D. 282
Garvey, C. 93
Gibson, V. 144
Gilbert, D. 768
Gligor, V. D. 165
Goldstein, A. 93
Graff, J. 631
Graubart, R. 459
Gray, III, J. H. 728
Gupta, S. 165
Hamilton, D. 9
Heaney, J. E. 359
Heberlein, L. T. 262
Heffern, S. 423, 467
Heller, H. A. 429
Hemenway, J. 282
Heffern, S. M. 236
Higgins, J. 523
Hosmer, H. 369, 409, 775
Housley, R. 799
Howe, D. M. 244
Humphreys, T 746
Jajodia, S. 57, 134
Jansen, W. A. 574
Earl Jenkinson 252
Howard L. Johnson 252
Michael V. Joyce 359
Ketley, C. 800
Keus, K. 800
Keys, J. 802
Klein, Y. 300
Krajewski, Jr., M. 76
Kramer, F. E. 236, 467
Kuhn, R. 554
Lambert, P. A. 799
Litchko, J. P. 802

Long, G. S. 761
Landry, J. C. 272
Lazar, R. 728
Lee, T. M. P. 448
Levitt, K. N. 194, 262
Liddle, A., LtCdr, Royal Navy 738
Looney, H. 738
Lynch, N. 768
Maconochy. W. V. 768
Madsen, W. 758, 770
Mandeville, L. 802
Mashayeki, C. 523
Maughan, D. 799
Mayer, F. L. 153
Mayhew, G. 310
McAuliffe, N. 184
McKenney, B. W. 620
Merritt, L. 754
Mildner, J. 754
Mukherjee, B. 262
Mundt, K. H. 438
Murray, W. H. 707
Nardone, S. 801
Nazario, N. A. 740
Nelson, D. 423,467
Neugent, W. 18, 203
Neumann, P. G. 28
Notargiacomo, L. 57
O'Neill, M. 544
Owen, Jr., R. W. 584
Padilla, S. J. 153
Peele, H. B. 111
Pierce, R., Capt, USAF 252, 389, 661
Pitcher, S. 768
Polk, W. T. 494
Price, W. R., Lt Col, USAF 292, 544
Price-Francis, S. D. 765
Proctor, N. E. 28
Puga, R. G. 564
E. Roche 300
Ross, R., LTC, USA 57, 754, 792
Sachs, J. 738, 795
Sandhu, R. S. 221, 329
Schell, T. 379
Schiller, C. A. 482
Schlipper, L. M. 399
Schwartz, M. 766
Slack, J. M. 513
Smith, G. 738
Smith, J. D. 534
Smith, M. 717
Solo, D. 799
Spafford, E. H. 641
Sterne, D. F. 680
Studt, J. 423,467
Suri, G. S. 221
Sutterfield, L. 66, 379
Swanson, M. 768
Taal, F. 300
Thuraisingham, B. 319
Toth, P. R. 84
Troy, E. 792
Unger, E. A. 513
Van Dulm, M. 300
Van Essen, U. 300
Varadharajan, V. 593
Walters, D. 744, 799
Weeber, S. A. 641
Wellham, G. 802
West, C. 774
White, G. B., Maj, USAF 66, 379
White, M. 799
Willcox, D. A. 697
Wilson, W. 738
Winkler, J. R. 272
Wolf, P. 300
Wood, J. 792
Yang, L. 473
Yates, J. 300
Yelton, M. 423,467
Zomback, R. 754

A MULTILEVEL SECURE DATABASE MANAGEMENT SYSTEM BENCHMARK*

Linda M. Schlipper, Jarrellann Filsinger, and Vinti M. Doshi
The MITRE Corporation, 7525 Colshire Drive, McLean, Virginia 22102

ABSTRACT

With the availability of various commercial multilevel secure (MLS) database management systems (DBMSs), performance evaluation tools will be necessary to assist users in understanding their performance characteristics. Benchmarking is one such performance evaluation tool. A number of benchmarking tools and methodologies have been developed for single-level databases, but these do not consider the effects of security-related factors like security level distribution, polyinstantiation, etc., in MLS DBMSs. In this paper, we describe an MLS DBMS benchmarking methodology that we have developed at MITRE, based on modifying the popular Wisconsin benchmarking methodology for single-level DBMSs. Currently the MLS DBMS methodology is limited to examining performance in a single user context, but work is ongoing to enhance it for the multiuser environment.

1. INTRODUCTION

Now that commercial multilevel secure (MLS) database management systems (DBMSs) are beginning to appear, potential users of such systems are looking for tools to evaluate their capabilities and performance for use in application environments. Performance, always an issue for DBMSs, becomes even more so for MLS DBMSs because of potential overhead associated with multilevel processing and various security options. Performance evaluation tools are needed to aid in assessing existing MLS DBMS technology and help identify areas where improvement is needed. Only some very preliminary work has been done in characterizing the performance aspects of MLS DBMSs [5, 7]. A good performance tool to evaluate MLS DBMSs still needs to be developed.

Two types of performance tools being used for single-level (conventional) databases are algebraic models and benchmarks. Algebraic models are a type of analytic model that can also be used as the basis for simulations. These models can be used to evaluate the performance of a proposed DBMS architecture before the system is actually prototyped, whereas benchmarks are experimental evaluation tools that can be used for performance evaluation after a system has been built. The model suggested by Mukkamala and Jajodia is an example of an algebraic model of MLS DBMS performance. It takes into account user behavior, system behavior and database characteristics to develop a performance model. (This model, however, focuses on a very limited aspect of performance: the effect of decomposition of multilevel relations on performance of the SeaView architecture [2].)

With the availability of commercial MLS DBMSs, systematic benchmarking and experimental validation of performance models is feasible. Benchmarking provides empirical measurements for comparing performance of different systems, developing real-world performance insights, or evaluating the accuracy of analytical performance evaluation tools. A systematic benchmark can also be used as a tool to evaluate new algorithms and query optimizers. Although there are several benchmark methodologies available to aid in the development and analysis of conventional DBMSs, they are not adequate for use with MLS systems because the effects of such security-related features as the number of security levels of data, polyinstantiation, or auditing options are not examined.

Thus far, benchmarking of MLS DBMSs has been performed only on an ad-hoc basis; for example, see [5]. A good benchmarking technique for MLS DBMSs is still needed to evaluate the effect on performance of security-unique factors such as the security level distribution (i.e., the number of security levels and compartments and the proportion of data associated with each) and user session level distribution. It is also important to determine the sensitivity of different MLS DBMS architectures to these and other security-related factors like polyinstantiation and auditing. At MITRE, we are developing a benchmarking methodology which can be used to give a general measure of the performance of various MLS DBMSs and can be further tailored to measure performance for specific application environments.

In this paper, we present our initial results in developing a generalized benchmarking methodology for performance analysis of MLS DBMSs in a single-user environment. The approach is to modify the Wisconsin benchmark [3], one of the most widely accepted benchmarks for conventional DBMSs. This benchmark is of particular interest not only because of its popularity, but also because of its use of a synthetic database. A major consideration in any

* This work has been done under MITRE core funding

benchmark experiment is the data used for testing. Although real data gives the most accurate results when evaluating a system for a known database application, it is often difficult to get, especially in the case of MLS DBMSs, which are designed for classified data. A synthetic database gives the opportunity to control security variables like security level distribution to test the effect of access mediation on performance. A synthetic database is also easy to tailor to mimic operational characteristics of a particular real-world database application, so that alternative products can be considered for a specific application environment.

The remainder of this paper is organized as follows. Section 2 presents an overview of security-related performance issues in MLS DBMSs. Section 3 describes the MLS benchmarking methodology and tools which have been developed. Finally, section 4 summarizes our results to date and discusses continuing work, including testing the methodology with currently available MLS DBMSs and enhancing it for the multiuser environment.

2. SECURITY-RELATED PERFORMANCE FACTORS

In this section we identify security-related characteristics that may affect the performance of an MLS DBMS. Although specific DBMS implementations will vary, some general observations can be made about the influence of these security-related characteristics on performance of various DBMS architectures. Before discussing the security-related performance factors, we briefly review the different basic types of architectures used in trusted DBMSs.

2.1 Secure DBMS Architectures

The major architectures being used for trusted DBMS products [5, 6] are the Trusted Computing Base (TCB) Subset, Trusted Subject, and Integrity Lock architectures. They differ in whether security responsibilities are allocated to the operating system (OS), the DBMS, or an intermediary between the user and the DBMS.

TCB Subset Architecture - In the TCB Subset architecture, a multilevel database is decomposed into single-level parts which are stored in separate OS objects. As shown in Figure 1, the MLS DBMS does not enforce the security policy, but relies on the MAC protections provided by the underlying trusted OS.

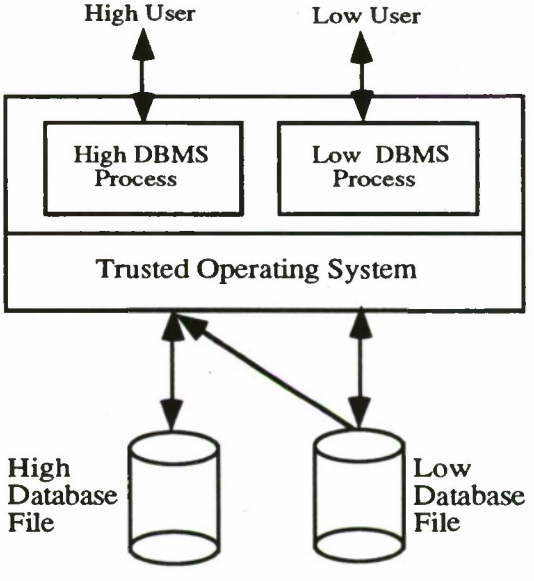


Figure 1. TCB Subset Architecture

Trusted Subject Architecture - In the Trusted Subject architecture, a security kernel in the DBMS handles both mandatory access control (MAC) and discretionary access control (DAC). The DBMS software runs on a trusted OS, and the multilevel database as a whole is stored in OS objects. But the DBMS associates security labels with DBMS objects and uses these labels as the basis for MAC. Figure 2 gives an overview of the trusted subject architecture.

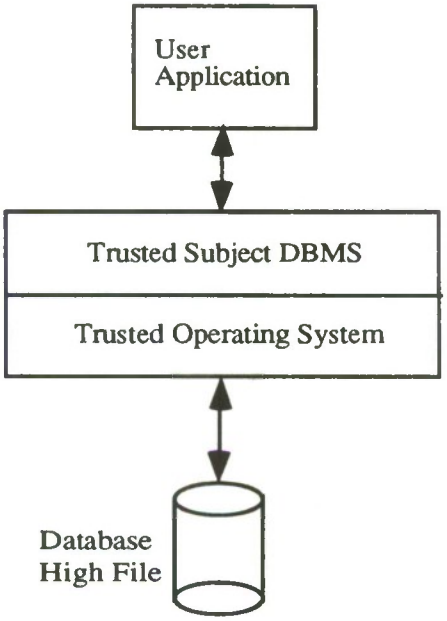


Figure 2. Trusted Subject Architecture

Integrity Lock Architecture - As shown in Figure 3, the Integrity Lock architecture uses a trusted filter to control access to data stored in an untrusted DBMS. The filter mediates all access between the users and the database. The filter is responsible for labeling data and restricting access to the data based on the user's security level. The labels are protected from modification while in the untrusted DBMS by a cryptographic checksum computed over the data and the label, which is also encrypted. On insert, the trusted filter computes the checksum and stores it with the data. On retrieval, the filter decrypts the security label and recomputes the checksum to confirm the data's associated security level, then determines whether the user is cleared to view the data before passing it on to the user's process.

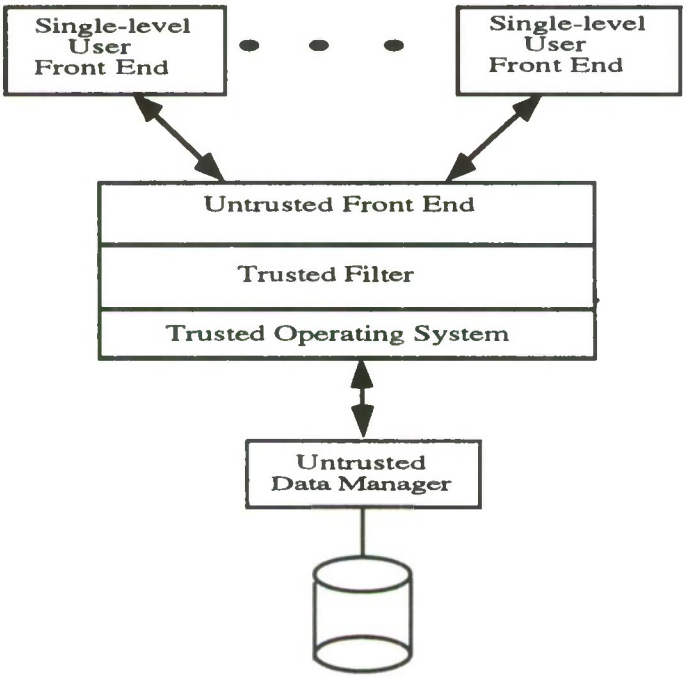


Figure 3. Integrity Lock Architecture

2.2 Security-related Factors Affecting Performance

In this section we discuss security-related factors and their possible effects on performance. These factors include access mediation, security level distribution, concurrency control, session level distribution, polyinstantiation, auditing, security constraints, and the Trusted Computer System Evaluation Criteria (TCSEC) [8] evaluation class. As in any complex system, the various factors are interrelated, and a system's overall performance reflects their interaction. Some tentative observations can be made about the performance implications of various security-related issues, but actual performance testing of MLS DBMSs needs to be done before drawing final conclusions.

Access Mediation - MLS DBMSs enforce two types of access control: MAC and DAC. To enforce MAC, many labels (e.g., file, relation, and multiple tuple labels) may need to be checked to satisfy a database access request. The TCB overhead associated with MAC checks for database access requests has obvious potential impact on the retrieval or update performance of an MLS DBMS, depending on its architecture. Object granularity can also affect the overhead of associated MAC checks to varying degrees depending on the architecture used.

Performance effects of DAC may depend upon the objects of DAC enforcement (e.g., relations vs. views). At lower TCSEC classes, DAC is not fundamentally different than the type of access control implemented in conventional DBMSs, and its performance effects are likely to be similar. However, DAC requirements in MLS DBMSs at class B2 and above (or future attempts to provide high assurance DAC) add to the complexity of the system and may have an effect on performance.

Security Level Distribution - The distribution of the classification levels among the data, in combination with the implementation architecture and the MAC object granularity, may have a significant impact on the performance of the MLS DBMS. Some potential effects of classification level distribution on the performance of various MLS DBMS architectures are as follows.

TCB Subset Architecture: In the TCB subset approach, the MLS DBMS relies on the MAC policy enforced by the trusted OS. To take advantage of the OS's MAC, the DBMS places data objects in OS data objects (e.g., files) with the same security level. When information is retrieved from the database, all files dominated by the user's session level may need to be accessed. If multiple security levels are being accessed, at least that many data pages must be read. Index entries may also be partitioned into multiple sub-indexes by security level. If a user is cleared to view all the data in the relation, then all of the sub-indexes must be examined, increasing retrieval time. [5] has suggested that with indexes partitioned by security level performance degrades roughly in proportion to the number of security levels. However, with few security levels or with most users operating at the lower levels, the performance impact of this approach would be less significant.

Trusted Subject Architecture: In the trusted subject architecture, the MLS DBMS performs access mediation and operates across different OS sensitivity levels. As a result, the database can be placed in one OS file. (Some trusted subject architectures, however, may still partition data among single-level internal data structures [9].) Although the trusted subject architecture may eliminate the need to separate data and indexes by security level, it increases the size and complexity of the TCB. Implementation decisions may be made to balance the requirements for a small TCB with performance features. Benchmarking on one prototype based on this architecture found performance impact when the security levels of tuples were different from those of the containing relations [5].

Integrity Lock Architecture: In the integrity lock architecture, a trusted filter generates cryptographic checksums for each tuple and its sensitivity label. Since the checksums must be recomputed for each tuple upon retrieval or update; performance is affected by the number of tuples returned to the trusted filter. The degree of impact depends on the speed of the checksum algorithm. However, since the data is stored in one system-high object such as an OS file or a backend DBMS server, all tuples that satisfy qualification criteria are returned to the trusted filter for access mediation. Thus, the time to complete a query may be independent of the user's session level. On the other hand, relative performance (the amount of time required per tuple returned) can vary widely based on security level distribution within a relation, since the number of tuples that must be examined and discarded depends on the proportion of data in the relation that is marked within the user's range.

Concurrency Control - Isolating users at low security levels from the activity of users at higher security levels is a problem that affects the mechanisms used for concurrency control and transaction management. For example, if locking methods are used, high users cannot be allowed to take locks that are detectable by low users. To avoid a covert channel, if a conflict arises, high users' requests must be delayed or aborted and low users' locks honored. Starvation is likely to occur, when low users frequently take write locks on low data and high users' read requests are

repeatedly aborted. Starvation can cause severe performance problems for high users, if there is contention for frequently used data. The performance of an MLS DBMS may be affected depending on the algorithm being used for concurrency control, the workload, the distribution of data, and the locking granularity.

Session Level Distribution - In the same sense that the security level distribution of the stored data can have an impact on performance, so can distribution of the session levels of the users accessing the data in a multiuser environment. With multiple users, session level distribution can be expected to affect performance through concurrency control, polyinstantiation, etc. These issues have yet to be thoroughly explored, since our initial work pertains only to the single-user case. With a single user, the performance effects of session level would be limited to those involving its relationship with the security level distribution of data being accessed, as noted previously.

Polyinstantiation - Polyinstantiation [2] allows a relation to contain multiple rows with the same primary key; the multiple instances are distinguished by their different security levels. For example, if a user whose session level is low attempts to insert a tuple with the same primary key as an existing high tuple, the low tuple is inserted nevertheless, with potentially conflicting data. The performance for inserts, updates, and deletes could be impacted by polyinstantiation. Retrievals might also incur additional overhead, since multiple instances of what would otherwise be one tuple may need to be accessed. Also, databases frequently updated by users at different session levels could grow larger, introducing performance considerations related to the size of the database. The security level distribution, and the update workload distribution, will affect the degree to which polyinstantiation and its attendant performance effects will occur in a particular environment.

Auditing - Auditing of security-related activities is required in MLS DBMSs. Security-relevant events include logins, granting and revoking of access permissions to relations, etc. The level at which auditing needs to be done is variable. The performance effects of the optional auditing features should be carefully examined, since their use may be a significant factor in the performance of data management functions. Audit queue management is another factor. An audit queue is a buffer for holding audit records until they are written to the audit log. If the queue is too small, performance can degrade while records wait to get into the queue; if it is too large, memory resources are wasted.

Additional Security Constraints - Context-based access controls, association constraints, and aggregation constraints are additional types of security constraints that may be addressed in future MLS DBMSs. Context-based access control restricts access to data based on whether it is viewed together with relevant context information. Association constraints classify the relationships between entities in the database. Finally, aggregation constraints specify that some aggregates of data have a higher sensitivity level than the individual components. These additional security constraints are outside the scope of current products (although TRUDATA [1] does have a mechanism to statically label certain predefined joins between two relations). However, researchers in this area are looking at rule-based (or knowledge-based) systems to implement these constraints. Overhead associated with this type of software as well as the number of constraints may be expected to impact the performance of the associated DBMS.

National Computer Security Center (NCSC) Evaluation Class - At higher NCSC TCSEC evaluation classes, not only are additional security mechanisms required but also the degree of assurance of the system's trustworthiness increases. To achieve higher degrees of assurance in the security of a system, the size of the TCB must be minimized, and the complexity of the security-relevant portions of a system's design must be reduced. With the introduction of higher assurance levels into an MLS DBMS comes a series of trade-offs the DBMS vendor must make with respect to performance and functionality.

3. MLS DBMS BENCHMARKING METHODOLOGY

The methodology that we developed for MLS DBMSs is a modification of the Wisconsin benchmark [3, 4], one of the most widely used benchmarks, and one which has been effective in uncovering performance and implementation flaws in the original relational products. The Wisconsin benchmark was developed as an application-independent benchmark, and consists of algorithms and a schema design for generating synthetic databases, a comprehensive set of queries for testing relational functions, and a methodology for systematically conducting single-user benchmarks. It does not yet include tests for concurrency control and recovery management. However, because of its use of a synthetic database and easy-to-understand structure of relations and queries, the Wisconsin benchmark provided a good starting point for development of an MLS DBMS benchmark. A synthetic database is of special importance for an MLS DBMS benchmark, because multilevel classified data is neither easily available nor easy to operate with. Also, a synthetic database permits systematic benchmarking with update and retrieval queries with specific selectivity factors and modeling of a variety of sensitivity level distributions, potentially a major performance issue in MLS DBMSs. Random generators can be used to obtain uniformly distributed attribute values for varying relation sizes.

This section describes the components of our MLS benchmarking methodology: the test database generator, which produces test databases modeled on those used in the Wisconsin benchmark; a generic query suite, which addresses relational functional coverage; and the query execution program, which provides the benchmark testing environment.

3.1 MLS DBMS Benchmark Database

We have developed a multilevel relation generator to generate the synthetic MLS DBMS benchmark databases. The relation generator was developed by modifying the algorithm for generation of Wisconsin relations [4]. Data values for each relation in an MLS database are generated using a C program that computes random integer values and corresponding string values to populate an MLS DBMS benchmark relation. The program is parameterized so that the number of tuples, security label formats, and security label distributions can be specified.

The relation generator is used to generate a database consisting multiple relations with similar schemas, which permits construction of join queries as well as other queries based on different selectivity factors. Relations with different sizes are generated to study the effect of relation size on performance using single-relation queries. The cardinality of the relations can be scaled relative to the computer platform's main memory capacity to reduce buffering effects upon performance of DBMS functions. [4] recommends that the largest benchmark relation be at least five times the size of the available main memory buffer space in bytes. Duplicate relations of the same cardinality can be added to this basic design to minimize the effect of buffering between the queries sequentially executed in a benchmark test. (But this may not always be desirable, since buffering and repeated execution of the same query may be a realistic reflection of some application environments.) The basic design can be tailored for specific application environments.

The output file of the relation generator is a flat, ASCII character file in a tab-delimited format that is suitable for loading with any DBMS bulk load utility. This approach supports development of database load and unload tests, a feature of many DBMS assessments. More importantly, the use of a bulk load approach facilitates use of the software with different MLS DBMSs, since the program does not interface directly with a DBMS. Our experience with B1-targeted MLS DBMSs indicates that bulk load utilities can be expected to be available with these products. Although ordinarily the security label for data entered under the control of an MLS DBMS is equal to the session level of the process writing the data, the bulk load utilities of B1-targeted MLS DBMSs permit a security label field to be included in the load file for multilevel loading of data. MLS DBMSs targeted at evaluation classes above B1 may not provide this multilevel load capability. However, the mechanisms for specific features such as bulk loading can only be speculated upon since no commercial B2-or-above MLS DBMSs are presently available. At least single-level bulk load utilities, though, will almost certainly be provided, since customers must be able to move data from existing DBMSs into the new systems. Only minor changes would be needed to adapt the relation generator program for use with a single-level load utility. It would simply need to write the generated data into multiple ASCII files destined for different security levels rather than into a single multilevel file.

Some of the features of the benchmark database design are as follows:

Relation Size - The benchmark database consists of a set of relations differing only in their cardinality (number of tuples). The smallest and largest relations should differ in size by a factor of 10, although their specific sizes can be scaled as we have noted previously. The example test database that we created for our initial benchmark consisted of five relations ranging from 1000 to 10,000 tuples.

Attributes - We added a security label attribute, *secLabel*, to the basic Wisconsin relation schema [4], to allow labeling of tuples within the test relations. In order to accommodate different label formats and different security level distributions, the format and distribution of data values for the *secLabel* attribute are specified by the user at the time a relation is generated. The other attributes contain 13 integer-valued attributes and three 52-byte string attributes which are the same as those defined for the Wisconsin relation schema. The two attributes, *unique1* and *unique2*, contain values in the range 0 to MAXTUPLES-1 where MAXTUPLES is the cardinality of the relation. One is random, while the other sequentially numbers the relation's tuples. The *two*, *four*, *ten*, and *twenty* attributes repeat in a cyclic pattern, but the values are randomly distributed in the relation by computing the appropriate modulus (mod) of *unique2*. The *onePercent* through *fiftyPercent* attributes simplify the scaling of queries according to certain selectivity factors. The string attributes, *stringu1* and *stringu2*, are string analogs of *unique1* and *unique2*.

Labels - The additional attribute, *secLabel*, is used to store tuple-level labels and to implement various distributions of security label values. Tuple-level labeling was selected because this is the MAC object granularity currently

supported by most MLS DBMS products. An allowable range of classification levels is associated with each relation in the database, and a security label attribute is associated with data in each tuple in a relation. The range and distribution of security levels may be changed in order to model different operational scenarios.

Storage Organizations - One of the key factors affecting the speed of a DBMS's retrieval operations is the storage organization of the relation. Three types of storage structures are used in the multilevel database: sequential organization, clustered index and nonclustered index. Although others exist, these were selected because they are supported by most relational DBMSs. In sequential organization, tuples are stored in the order they are entered. The time required to search a sequential file is long since all the records must be scanned. Retrieval time with a sequential organization is simulated in our benchmark by using non-indexed fields in the query selection criteria.

In a clustered index, the index determines the physical placement of data within a file. Typically, tuples are stored in sorted order based on the value domain of the clustering attribute or attributes. Although any index can allow quicker data retrieval by eliminating the need to scan the entire relation (provided index attributes have a high selectivity), clustered indexes provide one of the most efficient access methods for queries that access a range of key values. This is because the data resides on a smaller number of pages, requiring fewer physical page fetches by the DBMS to retrieve the qualifying records. In an MLS DBMS, however, the efficiency of a clustered index may be affected if tuples are partitioned by security level. For example, a tuple with a cluster key value of "10" may not follow one with a value of "9" if the tuples have different security labels. For the multilevel benchmark database, a clustered index is defined for each relation using the attribute *unique1* which contains a unique value for each tuple and whose values are generated in sort order for the load file. The security label attribute (*secLabel*) is also included as part of the index key. This is necessary in order to support polyinstantiation since the same "apparent" primary key may exist in tuples at different security levels. Since clustered indexes determine the physical storage location of tuples in a relation, only one clustered index can be defined for each relation in a database.

In a nonclustered index, the index is used to locate a pointer to a tuple or tuples containing the index key value. Since most DBMSs do not restrict the number of nonclustered indexes that can be defined for each relation, this type of file organization can be used for both secondary indexes and primary indexes. In the MLS database, nonclustered indexes are defined on the attributes *unique2*, *onePercent*, *tenPercent*, and *stringu1*.

3.2 MLS DBMS Benchmark Test Query Suite

We modified the test query suite of the Wisconsin benchmark to include queries that measure the effect of various security-related factors on the performance of the MLS DBMS. For completeness, we added sort queries, which are not included in the original Wisconsin query suite. The resulting test query suite contains selection queries, projection queries, join queries, aggregation queries, sort queries, and update queries. A brief overview follows.

Selection Queries - The speed at which a DBMS can process a selection query depends on the size of the relation, the storage organization of the relation, the selectivity of the selection criteria, the output mode of the query, the hardware speed, and the quality of the software. For a single-level relation, selectivity of a selection operation, σ, is the proportion of tuples that participate in the result of that operation. For an MLS DBMS, defining selectivity is difficult. In a multilevel database, a relation, R, is conceptually fragmented as $R_1, R_2, ..., R_n$ according to the distribution of classification levels of the tuples in the relation. From a user's perspective, R is restricted to the set of tuples R_u a user is authorized to access with clearance c_u and defined by

$$R_u = \{R_i \in R \mid \mathrm{cl}(R_i) \leq c_u\}, \text{ where } \mathrm{cl}(R_i) \text{ is the highest classification level of data in } R_i$$

Since $R_u \subseteq R$, the number of tuples returned by a selection operation may change with the classification level at which the query is executed. Therefore, selectivity is dependent on the session level. Performance analysis should examine both the case in which the number of tuples returned varies with the session level and the case where the number of tuples is held constant. Controlling selectivity is necessary to study the latter case.

Projection Queries - Although a projection may or may not contain duplicates, a projection query specifies elimination of duplicates. Most of the processing time for projection queries is incurred in eliminating duplicate tuples, usually by sorting. For a multilevel relation, it is difficult to determine the projection factor precisely because the correlation between projected attributes and their security levels may vary. Some MLS DBMSs will consider the security level to be a part of the projected attributes, even if it is not explicitly requested. Thus, the

cardinality of a multilevel projection may be larger than in a single-level database because an attribute value of "1" in an Unclassified tuple will not be considered a duplicate of an attribute value "1" in a Confidential tuple.

Join Queries - The join queries in the Wisconsin benchmark test how efficiently the system makes use of available indices and how query complexity affects the relative performance of the DBMS. In the MLS version of the query suite, the selectivity factor for join queries was reduced from 10 percent to one percent. For a 10,000 tuple relation, 10 percent selectivity generates a 1,000 tuple result, larger than would be desired by a user in an ad hoc environment. The definition of selectivity for joins of multilevel relations is the same as for multilevel selections.

Aggregate Queries - The aggregate queries for the multilevel benchmark are the same as in the Wisconsin benchmark. The queries use either a secondary index or no indexes. Indexed and non-indexed versions of an aggregate query are included to determine whether the query optimizers use an index to reduce the execution time.

Sort Queries - The Wisconsin benchmark query suite does not include any sort queries, but for completeness we included them in the MLS benchmark. They measure the ability of a DBMS to sort on alphanumeric or numeric attribute values. We have two versions of queries: one in which an indexed field is used to define the sort order and one which sorts on a non-indexed field. Although the second version more accurately reflects a DBMS's sorting capabilities, the versions using an indexed sort field are included for comparisons of index efficiency across MLS DBMSs with different security level distributions.

Update Queries - Update queries evaluate the overhead involved in updating each type of index when inserting, updating, or deleting a tuple. The update queries used in the Wisconsin benchmark address only single-tuple updates and are sequenced so that the database is restored to its initial state when all the tests are completed. Because the MLS update queries reflect the same limitations, they are probably the weakest part of the MLS benchmark. Not enough updates are performed to cause a significant reorganization of data pages or index nodes (especially for B-trees that are already several levels deep). A more realistic evaluation of maintenance overhead requires a multiuser environment, where the effects of concurrency control and deadlocks can also be measured. Additionally, the effect of polyinstantiation probably cannot be detected at the granularity of single-tuple updates, especially in a single-user test environment. Therefore, multiple-tuple update queries will be added when we develop a multiuser query suite.

3.3 Query Execution Tool

Benchmarking a database involves the use of a mechanism to submit queries to the target database and take timing measurements. The tool must perform this task reliably and consistently to get dependable results. There are two techniques for executing a benchmark. One technique uses ASCII script files containing the benchmark workload that are executed through an interactive SQL interface. The MLS DBMS Benchmark uses the second technique, which involves the use of embedded host language interface program calls within a programming language. Both techniques are valid; however, the second technique allows for more accurate timing measurements. We developed the Job Script Execution tool using the C programming language to run a series of benchmark tests against DBMSs over multiple platforms. A simple ASCII shell script might have been faster to develop, but not as flexible. The timing of a query should be performed as close as possible to the DBMS kernel interface in order to get the most accurate measurement. An ASCII script cannot capture the appropriate granularity of timing data from the DBMS. For instance, a shell script is unable to capture the timing detail for the first record returned from the database ("time-to-first"). A shell script running at the interactive SQL interface also takes additional resource overhead while executing the queries. On the other hand, shell scripts are very good for initially testing the query suite for errors. The tool we developed to execute the query script workload and analyze the timing output is described below.

Job Script Execution Program Description - The Job Script Execution tool serves two basic purposes: (1) to submit the job script workload to the target DBMS and (2) to set the OS clock and record the timing data for each query submitted to the DBMS. The program takes two input parameters: the query script file name and the name of the file to be used for the query results. The program prompts the user for the session level (classification and compartments) at which the benchmark test will be run. If the logon is successful, benchmark testing begins. Each query defined in the job script is submitted to the DBMS. The time is recorded before the query is submitted, after the first record is returned and after the last record is returned. When the last query in the query script is executed, a DBMS logoff occurs and control returns to the main unit.

The entire job script is submitted to the DBMS for a total of three times to obtain an average of the timing results. With each successive run of the Job Script Executor program, a DBMS logon and subsequent logoff are performed. The logon and logoff sequence permits each run of the query suite to find the DBMS in a consistent initial state.

The host machine is dedicated to running the benchmark with no other users active during testing, so that other activity on the host machine is not a factor in the timing results.

3.4 Security-Related Experiments

In this section, we describe the experiments that should be run using the benchmark database, test query suite, and query execution tool to examine the impact of security-related features on the performance of an MLS DBMS. Specifically, these experiments are designed to address the issues of security level distribution within a database, user session levels, and auditing. Not all of the security-related issues identified in section 2 have yet been at addressed. Since the commercially available MLS DBMSs are targeted at the B1 level, the effect of the NCSC evaluation level cannot be assessed. Similarly, neither high assurance DAC nor additional security constraints described earlier are relevant with the B1 MLS DBMSs currently available. Finally, the current MLS DBMS benchmark is a single-user test, hence performance impact of concurrency control, polyinstantiation, and session level mix cannot be measured until a multiuser benchmark is developed. The experiments discussed below are those that can be carried out using the initial version of the MLS DBMS benchmark tools.

Security Level Distribution Experiments - This portion of the benchmark methodology looks at the MAC mechanisms of an MLS DBMS by examining the effect of security level distribution on performance. By security level distribution, we mean the number of security levels of data stored in the DBMS and the percentage of data stored at each level. We approach this issue by holding the query workload constant and running a complete set of benchmark experiments over different versions of the multilevel database, with differing security level distributions. The tests are run with the MLS DBMS's security feature options, such as polyinstantiation or auditing options, either turned off or set at their minimum level.

We plan to use three database configurations for the generic multilevel benchmark with B1-targeted MLS DBMSs. The first configuration contains data at only one level. That is, all tuples in each relation are labeled at the DBMS's lowest security level, Level 1 (e.g., "Unclassified). The second version of the database has two levels of data in each relation, with eighty percent of the tuples labeled at Level 2 (e.g., "Confidential") and twenty percent labeled at Level 3 (or "Secret"). This distribution roughly models the operational environments for which B1 systems are appropriate. The third version of the database adds categories, which represent non-hierarchical classification levels, to model a compartmented-mode environment. Forty percent of the tuples in each relation are labeled at Level 4 (e.g., "Top Secret"); twenty percent at Level 4, Category A; twenty percent at Level 4, Category B; and twenty percent at Level 4, Categories A and B. In each experiment, the session level is database high, the highest security level of the data within the database (e.g., Level 4, Categories A and B, for the third version of the database).

This multi-database version approach increases the length of time to conduct benchmark experiments since databases must be dropped and rebuilt each time experiments are run using a different security level distribution. The alternative, though, is to have duplicate copies of each relation, with differing security level distributions. For example, there might be three 1,000-tuple relations: one with all tuples at one level, another with an eighty percent/ twenty percent security label distribution; and the third with the category distribution described above. Aside requiring more storage space, it is not clear that the single database approach provides an equivalent environment. For example, the metadata may be affected by the presence of additional security levels, as may be the actual storage of data values. Other side effects may also be present that could be difficult to detect or control.

User Session Level Experiments - The second type of experiment looks at the effect of the session level at which queries are run. Analysis of timing measurements will show how MLS DBMS performance differs for users running at different security levels. Therefore, in addition to executing a complete set of the basic workload for each multilevel version of the database, a modified workload is executed at different user session levels against each version of the database. The selection criteria are modified, if necessary, so that each query is guaranteed to return the same number of rows, regardless of the user's session level. Our definition of multilevel selectivity does not require this, but the total time required to complete a query also depends on the number of rows returned. Therefore, meaningful comparisons of response times across multiple user session levels cannot be made unless the size (cardinality) of the result rows is kept constant. The attribute *unique2* is used for the selection criteria because the security label value is correlated with the value of *unique2* when the test data is generated. For the databases described above, the workload is run at seven different session levels against each version of the database: Level 1; Level 2; Level 3; Level 4; Level 4, Category A; Level 4, Category B; and Level 4, Categories A and B

Auditing Experiments - The last type of experiment examines the effect of auditing. Overhead from auditing always has a detrimental impact on a DBMS's performance. In an MLS DBMS, foregoing the use of auditing will not be an

installation option. However, all systems will have a minimal set of audit operations that are required and another set of audit events whose use is optional. The experiments described above are all conducted with minimal auditing; the identification of the minimal set of audit events is included in the benchmark documentation. Specific testing to determine the impact of additional optional audit parameters is conducted using only one version of the multilevel database. Although experience with using this benchmark may indicate otherwise, it is not felt that the security level distribution of the data within the database is a factor in audit overhead. Therefore, audit experiments may be run using any one of the three multilevel versions. User session levels, though, may have an impact, so each series of tests using a specific set of audit parameters is run using at least two session levels: benchmark low (e.g., Level 1) and benchmark high (e.g., Level 4, Categories A and B).

4. CONCLUSIONS

This paper has presented the results of our initial work towards developing a methodology for benchmarking MLS DBMSs. We have categorized a set of security-related features of MLS DBMSs that may impact their performance. Benchmarking components, including the multilevel test database, the query suite, and a single-user test execution program, have been defined and implemented. Finally, we have described a series of benchmark experiments to assess the performance impact of MAC and auditing on B1-targeted MLS DBMSs.

When the project began, this effort was perceived as only requiring relatively straightforward modifications to the Wisconsin benchmark in order to adapt it to a MLS DBMS context. However, a number of unique issues associated with the multilevel properties of relational databases, such as the meaning of selectivity across multiple classification levels, have already been identified. There appears to be no clear consensus regarding the MLS extensions to the relational data model, so variations among MLS DBMS implementations can be expected from different vendors. Therefore, actual benchmarking experience using a variety of COTS MLS DBMSs will be necessary before an MLS DBMS benchmarking methodology can be completed.

At MITRE, we are working to extend the benchmark to support multiuser performance testing in order to assess the impact of concurrency control algorithms, polyinstantiation, and session level mixes. Meanwhile, we plan to test the single-user benchmark against some of the MLS DBMSs currently available. Advanced security constraints, such as context-based access control, association constraints, aggregation constraints, and enhanced DAC mechanisms, when they appear, will become targets of additional investigation.

REFERENCES

[1] *TRUDATA Model 3BBL Trusted Facility Manual*, Release 2.0 Revised 27 June 1990, Atlantic Research Corporation, Hanover, MD.

[2] Denning, D. E., Lunt, T. F., Schell, R. R., Heckman, M., and Shockley, W. R., "A Multilevel Relational Data Model," Proceedings of the 1987 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, 1987.

[3] DeWitt, D. J., Bitton, D., C. Turbyfill, 1983, "Benchmarking Database Systems: A Systematic Approach," Computer Sciences Department, University of Wisconsin, Madison, WI.

[4] DeWitt, D. J., 1991, "The Wisconsin Benchmark: Past, Present, and Future," *The Benchmark Handbook*, edited by J. Gray, Morgan Kaufmann Publishers, Inc., San Mateo, CA.

[5] Graubart, R., "A Comparison of Three Secure DBMS Architectures," Proceedings of the 3rd IFIP WG 11.3 Workshop on Database Security, September 1989, Monterey, CA.

[6] Hinke, T., "DBMS Trusted Computing Base Taxonomy," Proceedings of the 3rd IFIP WG 11.3 Workshop on Database Security, September 1989, Monterey, CA.

[7] Jajodia, S., Mukkamala, R, "Effects of SeaView Decomposition of Multilevel Relations on Database Performance," Proceedings of the 5th IFIP WG 11.3 Workshop on Database Security, Shepherdstown, WV.

[8] Department of Defense (DOD), 1985, *Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD.

[9] Varadarajan, R., October 1991, "An Overview of Informix-Online/Secure," Proceedings of the 14th National Computer Security Conference, Washington, D.C., pp. 701-703.

THE MULTIPOLICY PARADIGM

Hilary H. Hosmer
Data Security Inc.
58 Wilson Road
Bedford, MA 01730

ABSTRACT

This paper identifies some shortcomings in the TCSEC/TNI/TDI paradigm for multilevel secure (MLS) systems and summarizes requirements for an alternate paradigm. It describes the Multipolicy Paradigm, suggests shifts in thinking about MLS systems, and raises important multipolicy issues: policy conflict resolution, adding user security policies to commercial off-the-shelf (COTS) products, evaluating and certifying multiple policy systems, and passing sensitive data across policy boundaries.

INTRODUCTION

OVERVIEW

This paper consolidates and extends the results of our research into multipolicy systems[1 2 3 4 5] performed over the last two years under Air Force Electronic Systems Division sponsorship. The Multipolicy Paradigm permits a multilevel secure (MLS) system to enforce multiple, sometimes contradictory, security policies. Metapolicies, policies about policies, coordinate the enforcement of the multiple security policies. Policy domain codes on data indicate which security policies to enforce on the data, and multiple label segments supply the attributes needed for each policy.

The Multipolicy Paradigm permits natural modelling of the multipolicy real world. It permits possibly inconsistent security policies, such as confidentiality and integrity, to operate together. It may provide a vehicle for users to add their own security policies to a system without disrupting or invalidating existing evaluated policies. It may ease policy integration problems by preserving the original classification of data when data is passed across policy boundaries. Finally, if implemented in high-speed parallel processing architecture, it may improve trusted system performance.

Commercial applications include medical, financial, reservation, library, investigative and other systems that cross policy domains. Military applications include multiservice logistics, the multiservice Strategic Defense Initiative and Command, Control, and Communication (C3) systems in multinational battle theaters, like the Persian Gulf War.

RATIONALE

Integrating security policies on today's multilevel secure (MLS) computer systems is a difficult, sometimes impossible problem. When the security policies themselves cannot be integrated, the systems built to implement these policies cannot be integrated either. Sometimes the only way to solve impossible problems is to transcend them. For example, when Copernicus developed a new model of the planetary system with the sun at the center, his paradigm simplified planetary astronomy and initiated waves of discovery by others. Thomas Kuhn documents a number of these ground-breaking paradigm shifts in his book, *The Structure of Scientific Revolutions*.[6] Hoping for similar breakthroughs, computer security founder Dr. Willis Ware has called for a new MLS paradigm which will make networking and integration of MLS systems easier.[7]

Copyright 1992 Data Security Inc.

Although the USA standards for trusted systems call for a unified system security policy, Data Security Inc. proposes a new security paradigm based upon multiple, perhaps contradictory, security policies. Multiple security policies may be necessary if:

1. There is more than one security goal, such as privacy, confidentiality and integrity;
2. The system serves diverse constituents with individual goals and plans, such as the United Nations (U.N.), European Community (EC), and other federations;
3. The system is composed of separately evaluated pieces, such as an MLS Database Management System (Trusted DBMS) and MLS Operating Systems (Trusted OS).

THE CURRENT PARADIGM

The current US paradigm is based upon three standards documents described below: the TCSEC, the TNI, and the TDI. The current paradigm may evolve significantly because of the ITSEC, the 'harmonized' European criteria, and the Federal Criteria, a standards document focused on commercial system security, now under development at NIST with NCSC support.

The TCSEC. The Department of Defense Trusted Computer System Evaluation Criteria (TCSEC).[8] embodies the United States' security paradigm. The TCSEC prescribes a unified "system security policy" made up of subpolicies such as Mandatory Access Control (MAC) and Discretionary Access Control (DAC) which all cohere together to form a single system security policy. The unified policy drives the choice of security mechanisms and is the foundation of most assurance efforts.

The single-policy paradigm works well with stand-alone systems but causes problems when systems must be networked or combined and security policy integration is required. For example, when MLS products with slight variations in policies (such as Operating System (OS), Database Management System (DBMS), and user applications) must work together, there are usually policy integration difficulties as well as other interoperability issues [3]. The policy integration problems are even more evident when systems enforcing different policies, such as U.S.A. Department of Defense (DOD), North Atlantic Treaty Organization (NATO), and France, must interact and share classified data.

The TNI. The Trusted Network Interpretation (TNI)[9] of the TCSEC enlarges the single-policy paradigm so that multiple policies may coexist on computer networks. It permits each node on a network to have its own nodal security policy, but stipulates that the network as a whole must have an overall global network security policy which is the basis for evaluating the security of the network.

The TDI. The Trusted Database Interpretation (TDI)[10] addresses the problem of composing systems out of separately developed and evaluated trusted software products. The trusted computing base (TCB) of each separate component is called a TCB subset. Each TCB subset can enforce a different security policy, such as MAC or DAC. The TDI assumes, however, that these subpolicies cohere into a single consistent overall security policy.

The ITSEC. The draft International Technology Security Evaluation Criteria (ITSEC)[11] permits a user to specify a security policy, select a system meeting site needs, then request a certified evaluation center to do an evaluation to provide the necessary assurance that the selected system is able, in fact, to carry out the user's security policy. There is no restriction on what functionality could be in the user's policy. The policy could include integrity, availability, non-repudiation, and confidentiality, for example. The ITSEC follows the TCSEC lead in requiring users to integrate multiple separate policies into a single coherent system security policy.

Problems With The Current Paradigm. The paradigm of a unified security policy has some major shortcomings which are becoming apparent as multilevel secure systems are fielded. 1. *It's inflexible.* If a user wants to modify built-in aspects of the system security policy, the whole system must be reevaluated. *2. Exchanging sensitive data with systems with other security policies is difficult or impossible in real-time.* Guards are needed at all interfaces, and mapping rarely can translate security

levels from one policy to the other without upgrading. 3. *It's unrealistic.* The real world has multiple coexistent security policies. A computer security officer creating an automated security policy[12] must often integrate diverse and contradictory security policies together into a single coherent policy to meet TCSEC criteria. Canada's experience trying to integrate the national privacy policy with the national disclosure policy into a single policy lattice illustrates the real difficulties users face [13]. 4. *Performance is poor.* Adding security to existing systems seriously slows down throughput.

The current paradigm must be enlarged or shifted to meet the needs of a more interrelated and integrated world. With a few significant enhancements, the single-policy paradigm can be extended into a more flexible, more interoperative, better-performing multipolicy paradigm.

REQUIREMENTS FOR A NEW PARADIGM

What must a larger and more inclusive paradigm do? It should:

Handle COTS system construction. Facilitate the integration and tailoring of commercial off-the-shelf products to meet the end-user's system security policies.

Separate the policy from the enforcement mechanism. The system security policy should not be such an integral part of the system that it is impossible to change policies without reevaluation.

Ease sharing data with other policy systems. In multinational conflicts like the Persian Gulf, US DOD users need to share classified data with allied computers that implement different service, national or International security policies.

Enforce the originator's security policy. Current strategies for sharing data across security policy boundaries (Guards, Man-in-the-loop) frequently must upgrade or downgrade data, thus losing the original classification. Even if the multinational situation is one of cooperation rather than conflict (for example, divisions of a multinational corporation, or international electronic funds transfer), it is desirable to guarantee enforcement of the originator's security policy while sharing data across computer systems.

Permit contradictory policies to operate in parallel. For example, different states have passed different laws about releasing AIDS data. If an AIDS patient from Connecticut is in a New York hospital, which state's disclosure laws should apply to the release of data? In the European Community health system, the varying disclosure laws of 12 different countries must be implemented and maintained.

Improve the performance of trusted systems. Adding security to a system usually degrades its performance significantly, largely because of auditing and access control checks.

Other. The list above is not exhaustive. As more multilevel systems are implemented, we will become aware of more difficulties and requirements. Solving these problems is essential to widespread user acceptance of MLS systems.

RELATED WORK

Many researchers have addressed aspects of these problems. Biba[14] and Clark and Wilson[15] established the importance of integrity policies. DEC built a multipolicy operating system SEVMS that enforces both confidentiality and (Biba) integrity, showing the commercial feasibility of multiple policy systems[16]. The European Computer Manufacturers Association (ECMA) developed a conceptual framework for security across multiple domains with multiple authorities[17], raising hopes for international standards.

Multiple policies frequently conflict. Dobson and McDermid discovered that Integrated Programming Environments (CASE tools) require three distinct security policies , and "it is critical to articulate and resolve the policy conflicts."[18] Dobson has been studying organizational security policies, the source of many conflicts. Trusted Information Systems (TIS) documented that the Aegis Combat System requires three sometimes conflicting policies (information disclosure, information modification, weapon release),

while the Nuclear Command, Control, and Communications system requires four policies (weapon release, denial of service, information disclosure, and information modification)[19] Secure Computing Technology Corp. (SCTC) and Georgia Tech Research Corporation (GTRC) in their Assured Service policy work identified ways that various availability mechanisms both complement and conflict with secrecy policies.[20] Rae Burns raised the inherent secrecy/integrity conflict[21] and Oracle Corp. addressed methods for resolving it.[22] Tradeoffs between competing policies are often required, and often only the ultimate users can determine which policy to emphasize. In a multipolicy environment, it is critical for users to be able to specify how policy conflicts should be resolved.

Multiple policies may be more complex than single policies. To manage this complexity, researchers are studying the fundamental properties of policies. Feiler and Dowson explored the relationships between policies and processes, discovering that policies may conflict and that policies about policies may be necessary.[23] Moffet and Sloman explored management policies and the need for explicit control authority in the commercial arena. They also explored how to represent and manipulate policies and came to view policies as objects which can be created, destroyed, queried; and which can interact with each other.[24] The Policy Workbench project at George Mason University(GMU) studied intentions implicit in policies, incompleteness in assumptions underlying security policy models, and ways to represent security policies, including activity role charts, Petri nets, data flow diagrams, and structural diagrams.[25]

Several researchers aim for policy flexibility. Grenier, Hunk, and Funkenhauser differentiated policies from mechanisms.[26] The Planning Research Corporation (PRC) proposed rule-based policies as a way to escape the inflexibility of built-in policies and demonstrated that assorted rule-bases can be plugged into the same system.[27] MITRE's General Framework for Access Control (GFAC) group asserted that all policies can be expressed as rules specified in terms of attributes and other information controlled by authorities.[28]

Our earlier work introduced several concepts which are incorporated into this paper. [1] proposed a Multipolicy Machine which enforces multiple, possibly contradictory security policies using Metapolicies, a term introduced in [3] and expanded in [2]. [4] proposed shared labels to save space, and parallel processing of policies and policies on ROM chips to improve performance and standardization.

THE MULTIPOLICY PARADIGM

COMPONENTS

Most security models built after Bell and LaPadula's classic model[29] include:

1) Subjects
2) Objects
3) Security Policy
4) Sensitivity attributes for subjects and objects
5) Policy Enforcer to mediate subjects' operations on objects in accordance with the policy.

Several additional components are required to handle multiple policies:

1) Multiple security policies;
2) Multiple security policy enforcers;
2) Multiple policy coordinators (metapolicies);
3) Assignments to specify which policies apply to which subjects and objects.

Two optional components are needed to provide flexibility and performance:

4) A means to control policy changes and updates;
5) A design to avoid policy-enforcement processing bottlenecks.

Each component is described below. We describe abstract concepts, then suggest concrete ways to implement those concepts.

MULTIPLE POLICIES

A **policy** is a set of constraints established by an accepted authority to facilitate group activity. A policy may be explicit or implicit, broad or narrow in scope, mandated or optional. **Security policies** are those policies whose goals are protecting the confidentiality, integrity, and/or availability of people, resources, and information. **Automated security policies** [11] protect information within computer systems, and require security policies that are much more explicit and formally specifiable than policies intended for people. Automated security policies typically include: a) definitions of subjects and objects; b) definitions of allowable operations; c) policy rules, and d) data for implementing the policy, such as a lattice for ordering sensitivity levels, integrity levels, compartments, etc. Automated security policies must be tamperproof and are, by definition, part of the trusted computing base.

In the Multipolicy Paradigm, a computer system can enforce more than one policy. The Multipolicy Paradigm permits multiple:

1) Types of security policies (Eg. Integrity, MAC, DAC, Chinese Wall[30]);
2) Variations of security policies (Eg. integrity by Biba and by Clark- Wilson);
3) Combinations of policies (Eg. hierarchical, independent, coordinated);
4) Sources of policy (Eg. user, administrator, government, standards body);
5) Means of changing policies (Eg. locally, remotely, at sysgen).

Unlike the current TCSEC paradigm, the Multipolicy Paradigm does not require that a unified system policy be developed, or even that the policies be consistent. Canada, for example, can implement separate privacy and confidentiality policies on one system[13], and the EC may keep several separate health and financial information privacy policies.

Policies in current systems are usually implemented as instructions in code, with instructions in the kernel TCB for system security policies and instructions in applications programs for user policies. In the Multipolicy Paradigm, complex data structures will be needed to implement each policy and its associated metapolicies. This implementation method can provide both flexibility and assurance. Examples are in the companion paper printed in this proceedings, "Metapolicies II"..

MULTIPLE ENFORCERS

Security Policy Enforcers Security policy enforcers implement the rules of a policy on the subjects and objects within the policy domain. Each enforcer is trusted to protect and enforce the policies in its domain correctly and must be tamperproof. Enforcers may be implemented in several ways. However, it is critical that the policy NOT be built into the enforcer, as it now is in most reference monitor implementations. One enforcer may enforce multiple independent policies, or multiple policy enforcers may enforce multiple different policies, or multiple versions of the same policy, or multiple subsets of the same policy.

Metapolicies Metapolicies are policies about policies. They provide a framework for clarifying policies, explicitly stating the assumptions about policies and the organization's control process for policies. They also coordinate the interaction between policies, explicitly specifying order, priority, and conflict-resolution strategies. Metapolicies clarify underlying policy assumptions and relationships, facilitate expression of the variety, richness, and multiplicity of security policies, and permit the controlled interaction of policies and subpolicies, making complex policy systems possible [2]. Metapolicies specify who can set policy, who can change policy, and the procedures for changing policies. They also include rules about developing, verifying, and protecting security policies and rules about the interaction of multiple security policies, especially where they conflict. The Multipolicy Paradigm permits multiple distinct security policy domains, administered by different organizational entities each with complete

policy autonomy in its domain, to be modeled in a computer system. Metapolicies control the interactions of the multiple policies.

MULTIPLE DOMAINS

A policy domain is a logical construct defining the area of responsibility of an authority. The U.S. federal government, for example, takes responsibility for regulating interstate commerce (the federal domain), while the states take responsibility for regulating intrastate commerce (the 50 state domains). NCSC, OSI, ISO, ECMA, DOD, and NATO are a few of the well-known security domain authorities.

Each security domain may be autonomous, with its own authority, subjects, objects, policies, and policy enforcement mechanisms. Others may be part of a hierarchical structure, like Air Force Base (AFB), Air Force System Command(AFSC), Air Force (AF), and Dept. of Defense(DOD). In hierarchical structures, the authority and policies of the top domains must be incorporated by the subordinate domains. Under the unified-policy model, the base, system command, AF and DOD policies would be integrated and implemented as a single automated security policy. However, under the Multipolicy Paradigm, each of the individual policies in the hierarchy - the DOD policy, the AF policy, the AFSC policy and the AFB policy - would be separate policies, and a policy domain code would be required for each. This gives the AFB security administrator the flexibility to change local base policy while leaving national DOD and AF policies untouched.

Domains may overlap each other, so that subjects or objects may belong to more than one domain and fall under more than one policy. Patients who fall under the confidentiality policies of multiple states, and military information which comes under both national and international confidentiality policies are members of overlapped domains.

Policies in different domains may conflict. However, there must be means to resolve the conflicts as they occur. For example, if a national and an international policy are in conflict, which takes precedence? A later section addresses conflict resolution techniques. Policies within the same domain must not conflict, because logical inconsistencies may create exploitable holes. Research is needed to see if policy conflicts are possible between subdomains.

POLICY ASSIGNMENTS

There are several ways to assign policies to data. The European Computer Manufacturers Association (ECMA) [17] has proposed security domain codes on security labels which indicate under which label convention the label is formatted, eg. International Standards Organization (ISO). We propose security policy domain codes as a mechanism to indicate which policy domains apply to this subject, object, or policy. Whenever policy decisions are made, these policy domain codes would be checked first so that the proper policy enforcers can be invoked. Figure 1 illustrates domain codes incorporated into security labels.

Single Policy Label Format:

OBJECT / SECURITY ATTRIBUTES / POLICY DOMAIN CODE

Single Policy Example:

The string of bits representing patient Jones' data release permissions will be interpreted in accordance with the New York privacy policy.

Patient / John Jones / 100101 / Privacy-NY

Figure 1A. Single Policy Domain Code Example

Multiple Policy Label Format:

OBJECT / LABEL / POLICY / LABEL / POLICY / etc.
/ SEG / CODE / SEG / CODE /

Multiple Policy Example:

Patient Smith, who lives in Connecticut, is hospitalized in New York and then sent for consultation to a teaching hospital in Massachusetts. The privacy policies for all three states apply to him and his hospital record has three sets of privacy attributes.

Patient / Sam Smith / 01011 / Priv-MA / 01010 / Priv-CONN / 11010 / Priv-NY

Note that labels with multiple policy attributes and multiple security domain codes may get very long. A paper published last year, "Shared Sensitivity Labels"[4] describes an indirect addressing technique which permits subjects and objects with the same sensitivity levels to share a single version of the label.

Figure 1B. Multiple Policy Domain Codes in Trusted Labels

MULTIPOLICY ISSUES

CONFLICT RESOLUTION

Strategies for resolving conflicts between policies include:

Resolve the conflicts manually and automate the integrated results. This is the strategy taken by most vendors and most user organizations. The information security officer manually integrates multiple, possibly contradictory policies into a coherent system security policy. This is a difficult process, since each policy has its own source or owner, its own enforcement authorities, and its own evolutionary time frame. Developing consensus takes a long time, especially if policies reflect deeply held values.

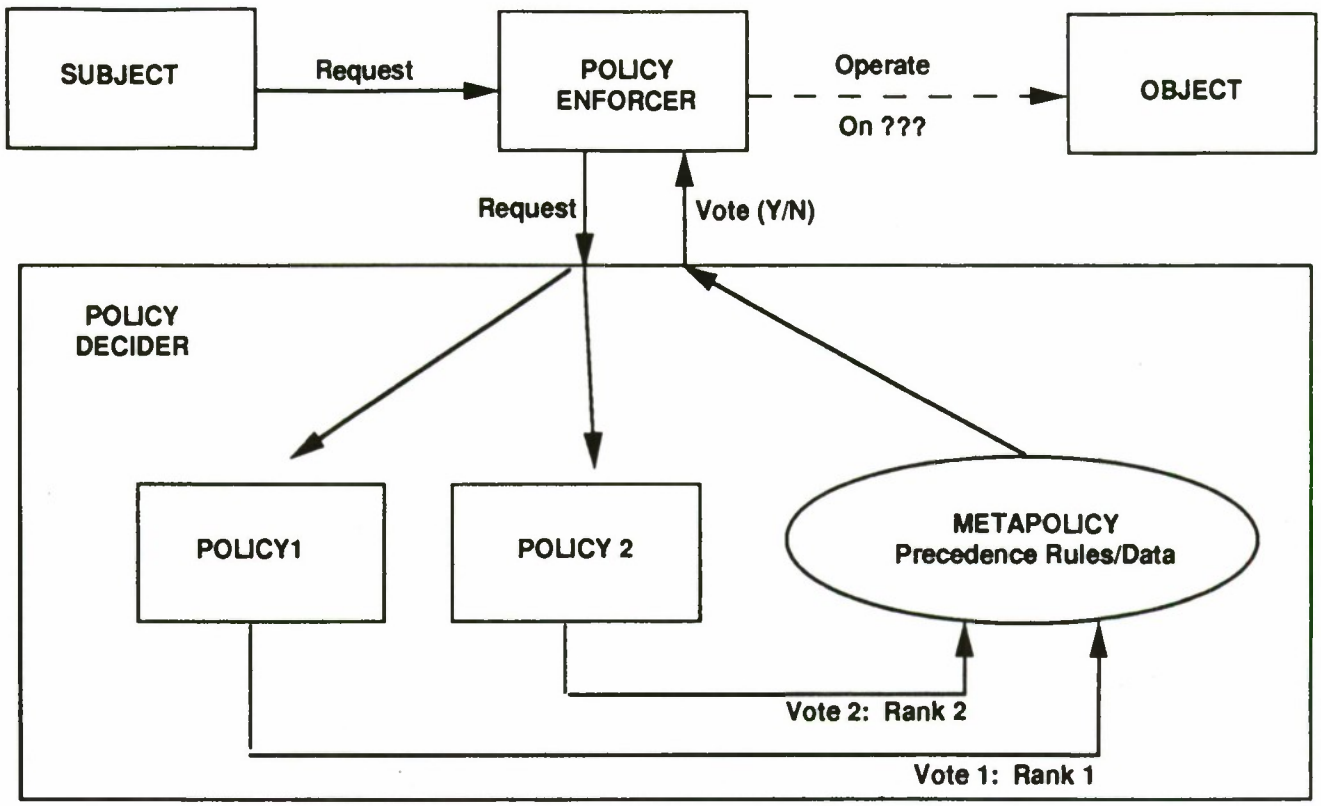
Resolve by dominance. If the policies are hierarchically structured, then the policy higher in the hierarchy predominates. If the policies are ranked by their importance, the most important predominates. Or, if the policies reflect the ranking of the authorities who created them, then the policies of the dominant authority predominate. This strategy is appropriate in the military and other hierarchically-structured organizations.

Translate policies into a common form. Dr. Bell advocates this strategy using policy conversion logic on a Universal Lattice Machine. He showed that multinational sharing, Clark and Wilson, dynamic separation of duty and ORCON can all be implemented with the Universal Lattice Machine.[31]

Run in separate policy domains. John Rushby's Separation Machine, as implemented by Amdahl's Multiple Domain Facility, allows seven different policies on one machine but no communication between domains.[32] Parallel processing of policies is possible but resolving conflicts between policies must be done outside the Separation Machine.

Use additional enforcement mechanisms to implement custom user policies in addition to DAC, MAC , etc. Type enforcement like that implemented in SCTC's Logical Coprocessor (LOCK)[33] provides considerable user flexibility.

Figure 2. Policy Conflict Resolution[34]



Policy Conflict Resolution in Figure 2

1) The 'Subject' wants to operate on the 'Object', but the request must be mediated by the 'Policy Enforcer'.

2) The Policy Enforcer passes the request to the 'Policy Decider' along with the subject and object policy domain codes. The Decider consists of multiple 'Policy Decision-Makers' operating in parallel, one for each policy implemented by the system.

3) Based upon policy domain codes, the request is routed to the proper Policy Decision-Makers.

4) Using rules and decision data to evaluate the request, each Policy Decision-Maker sends its Policy Precedence Ranking and a Vote (eg. Yes', 'No', 'Don't Care', 'Undecided' or a fuzzy logic number on a continuum) to the Metapolicy.

5) The votes of all the individual policies (Vote 1 and Vote 2 in this example) are combined by the Metapolicy and weighed according to its rules as well as the precedence ranking of each policy (Rank 1 and Rank 2 in this example).

6) The resulting 'Yes' or 'No' vote is sent back to the Policy Enforcer which then permits or denies the requested operation.

Use rule-based access control. John Page[27], Marshall Abrams[28], Leonard LaPadula[35], and others' showed how rule bases (rules with one-to-one correspondence with the operations of the system) handle many kinds of access control policies. LaPadula proposed a voting technique to resolve rule conflicts which we adopt in Figure 2.

Adjudication. In case of conflict, develop a solution which reflects the tradeoffs and weights of the users on the system. If there are multiple applications which weigh things differently, accommodate the various weights. The use of metapolicies, or 'policies about policies', to sort out precedence and to identify and resolve policy conflicts is illustrated in Figure 2.

Outside mediation. When two security policies contradict each other, the decision about what to do may be best left to a human who understands the content and the context, as in downgrade decisions.

A combination of these techniques can be powerful. Figure 2 shows that the conflict resolution process can be simple and elegant, no matter how many different policies are included. If parallel processors are used to implement multiple Policy Decision-Makers [1], the decision-making time could be kept close to that of a single-policy machine.

FLEXIBLE USER SECURITY POLICIES

One of the frustrations experienced by users is their limited ability to modify the security policies which are built into COTS products. Most current systems allow changes to the policy data (eg. the contents of the lattice), but not to the policy rules. The problems are:

1. The rules are built in.
2. Assurance depends upon a stable policy.
3. Changes may introduce security flaws.

Although inconceivable with today's built-in policies, user authorities should be able to add or delete policies from their systems at any time. Standardized policies and labels may be distributed on ROM firmware [4] or protected software modules and would include metapolicies which describe the policies and their interrelationships. Metapolicies which coordinate the policies must be customized to the user's needs when each policy is installed.

There are several ways to provide policy flexibility. COTS vendors can offer customers a set of evaluated policy options, clusters of commonly-desired combinations, to choose from when the product is ordered. Trusted software can be used to tailor policies further. Modifiable aspects, such as label size, number of compartments, and which policies are selected for enforcement, must be carefully limited to maintain the integrity of the evaluated system. The vendor could tailor the system before shipping, or the System Security Officer (SSO) could tailor it at system generation.

Currently, user policies are coded many times into applications programs. It is desirable for integrity and control to get the policy out of the application and into the system where the same policy can be invoked by many programs. Ideally, an SSO should be able to enter entire user policies via trusted software into an isolated area of the TCB where their interactions with applications programs and other policies are carefully mediated by the appropriate metapolicies. It is clear that this method works when the user policies are a subset of the underlying system policy (TCSEC paradigm). It is unclear what should happen when there is no underlying system policy (Multipolicy paradigm).

Options which don't affect the security of the system, such as audit policy options and default options, can be set by the SSO at any time. The vendor should ship any trusted system with conservative defaults selected to err on the side of caution.

When policy change is required, domain administrators can implement changes in policy in their domains in a variety of ways. On automated systems, they can "securely download" new policy

modules, send new firmware chips for installation by each System Security Officer (SSO), or simply give orders to the SSOs to make changes. Metapolicies will restrict policy changing to authorized personnel.

The capacity to absorb multiple user policies (representing multiple nations, multiple divisions, or several kinds of integrity policies) without reevaluating the whole system is an integral part of the Multipolicy Model. However, evolution of policies raises the issues of reevaluation and recertification.

EVALUATION AND CERTIFICATION ISSUES

How does one evaluate and certify a system with multiple flexible policies? If policies change, whether at sysgen or on-the-fly, when must the system be reevaluated or recertified? There are many questions and problems.

Today, evaluators determine whether or not a system correctly implements a policy with what degree of assurance. The Multipolicy Paradigm requires a shift in thinking so that evaluators determine that the system has mechanisms which will correctly implement whatever policy it is given. The evaluators will examine the mechanisms for interpreting and enforcing a variety of policies rather than just one.

Evaluators cannot determine in advance that every possible policy which might come along will work correctly in the system. Certifiers will study the installation of specific policies on a specific system. Certifiers will also check the metapolicies which are set by the user, such as those that prioritize policies in conflict, Certifiers will need to check the interaction of multiple security policies and to handle the problem of too many possible combinations to test them all.

If evaluators evaluate systems separately from polices, who evaluates the policies? Since there are likely to be so many different policies, commercial evaluation centers, like the ones doing evaluations in Europe, would be appropriate for policy evaluation. If policies have been developed in different places by different authorities and evaluated in different places with different levels of assurance, common standards need to be developed. For example, for any desired certification level, each policy must have the minimum level of assurance for that certification level. Metapolicies which describe and control the policies must be evaluated as well.

CROSSING POLICY BOUNDARIES

Crossing policy boundaries is one of the most difficult problems in trusted computing. When classified data leaves one policy system, such as the US Military, to go to another, such as NATO, a trusted person or process must sanitize and relabel the data and approve the transfer. In the civilian world, privacy laws require that the patient or the patient's guardian give written approval for the transfer of medical data to another hospital. In both military and civilian life the data owner wants assurance that the data will be treated in the new policy realm in sufficient accordance with the owner's policies.

In a multipolicy system, it is possible for an object to go from one multipolicy machine to another without leaving its original policy domain. There are several important assumptions:

1. The Multipolicy Machines follow standards for labelling objects which preserve the integrity of labels and policy domain codes.

2. The Multipolicy Machines follow standard policies about handling objects from different policy systems. For example, if the label is checked and it doesn't match any policy in the system, the object is inaccessible to any users on the system. However, the inaccessible object may be passed on intact to another system which follows the standards for multipolicy systems.

The sending and receiving systems may implement a homogeneous, an overlapping, or a heterogeneous set of policies. As long as the receiving system is trusted to implement the policies indicated in the label associated with the object, there is no boundary-crossing problem.

If the receiving system does not enforce the policy or policies marked on the object, it must either pass the object on to another machine which enforces the appropriate security policy, hold on to the object without permitting any access, or. as is done now, request human intervention. Which choice is made could depend upon instructions which accompany the object, or on the metapolicy for the receiving computer.

IMPLEMENTATION OPTIONS

Throughout this paper we have suggested several reasonable approaches to implement a Multipolicy Paradigm:

1. Multiple sets of rule-based policies, as seen in Figure 2, [27] and [35];
2. Multiple co-processors, like SCTC's LOCK and Sidearm [33];
3. Distributed processors: each node has a local policy and a master node has them all; [1]
4. Parallel processors or policies in ROM chips to improve performance [4];
5. Multidomain machines, like Amdahl's Multidomain Facility [32];
6. Hybrids of the above

Other approaches are possible as well, but we do not wish to focus here on implementation options. More implementation option information appears on our "The Multipolicy Model; A Working Paper" [5].

APPLICATIONS

The Multipolicy Paradigm is useful whenever multiple security policies are involved, especially when normal security goals are extended beyond DOD confidentiality to include privacy, availability, integrity, weapons release control or other policies and wherever users with different values and tradltions must share a common system.

Military multipolicy applications include: multinational battle management, multinational command and control centers, logistics involving multiple services, and multinational communications systems. The Strategic Defense Initiative is a classic case of multiservice policy interaction, as was the Persian Gulf War. An application common to ordinary military systems would be to define peacetime, threat alert, and wartime security policies and shift from one to the other, rather than 'loosening' the peacetime policy[20] when war starts.

There is no single standard security policy, like that of the DOD, in the commercial world, so a trusted system, to be marketable, must be able to adapt to multiple policies. Although the TCSEC unified policy paradigm can adapt to a wide range of needs [Bell, 31], the Multipolicy Paradigm will facilitate expression of users diverse, unanticipated, and contradictory security policies.

Commercial applications for multipolicy machines are numerous. Multinational banks, multinational corporations, international non-profit activities such as the Red Cross and CARE, merged corporations with multiple corporate cultures, colleges and companies which cross state borders, International telecommunications systems, are all candidates for multipolicy systems.

In non-military government sectors there is even more potential for the multipolicy paradigm. Almost every system developed by the European Community needs multiple policies to express the different values and varying traditions of the nations involved. For example, a multipolicy international health system that permits different nations to control security policies for their own citizens is more practical than requiring twelve nations to come up with a unified confidentiality and privacy policy.

CONCLUSION

This paper identified shortcomings in the TCSEC/TNI/TDI paradigm for multilevel secure systems and summarized some of the requirements for an alternate paradigm. It briefly described other researchers' work in the area, then wove many contributions into a Multipolicy Paradigm.

The Multipolicy Paradigm supports multiple, perhaps contradictory security policies and has many applications and uses. Multiple contradictory security policies may be necessary if:

1. There is more than one security goal, such as privacy, confidentiality and integrity;
2. The system serves diverse constituents with individual goals and plans, such as the EC;
3. The system is composed of separately evaluated pieces, such as MLS DBMS and OS.

Multiple policy systems will be more flexible, but much more complicated in many ways than single policy systems. The paper addressed strategies for solving many of the key multipolicy issues. This feasibility study showed that:

Policy conflicts can be resolved;

Changes in ways of thinking are needed to evaluate and certify flexible multipolicy systems.

There are many strategies for getting policy flexibility while preserving assurance.

Users can add user security policies to commercial off-the-shelf products.

Multipolicy systems may ease the old problem of how to pass sensitive data across policy boundaries.

The Multipolicy Paradigm can be successfully implemented in many ways.

The Multipolicy paradigm will provide greater flexibility for users who need to add their own security policy specifics to the security policy of an existing system. It will make it easier to transfer data to systems in other security policy domains. It will let users model complex real world security policies more easily and permit contradictory policies to operate in parallel. Parallel processing may permit an improvement in trusted system performance, as well.

The Multipolicy Paradigm is now just a concept with potential. Much more work needs to be done to make it a reality.

ACKNOWLEDGEMENTS

This work was produced under a Small Business Innovative Research grant, contract number F19628-91-C-0157, under the sponsorship of the Electronic Systems Division of the Air Force Systems Command, Hanscom Air Force Base, Bedford, MA.

Several colleagues critiqued the multipolicy ideas as they evolved and contributed helpful suggestions: Marshall Abrams, Victoria Ashby, David Bell, Rae Burns, Dorothy Denning, Jon Graff, Tom Haigh, Grace Hammonds, Jody Heaney, Eric Leighninger, Bret Michael, and Bhavani Thuraisingham .

REFERENCES

[1] Hosmer, Hilary H. "The Multipolicy Machine: A New Paradigm For Multilevel Secure Systems", *Proceedings of Standard Security Label for GOSIP, an Invitational Workshop*, April 1991, NISTIR 4614, June 1991.

[2] Hosmer, Hilary H., "Metapolicies I", ACM SIGSAC Data Management Workshop, San Antonio, TX, December 1991, *ACM SIGSAC Review 1992*.

[3] Hosmer, Hilary, "Integrating Security Policies", *Proceedings of the Third RADC MLS DBMS Workshop*, Castile, NY. June 1990, MITRE Technical Paper MTP 385.

[4] Hosmer, Hilary H., "Shared Sensitivity Labels", *Database Security, Status and Prospects*, North-Holland, 1991.

[5] Hosmer, Hilary H. "The Multipolicy Model, A Working Paper", *Proceedings of the Fourth RADC Workshop on Multilevel Secure Database Systems*, Little Compton, Rhode Island, June 1991,

[6] Kuhn, Thomas, *The Structure of Scientific Revolutions*, 2nd Edition, University of Chicago Press, Chicago, 1970.

[7] Ware, Dr. Willis, on Computer Security Panel, AFCEA Conference, Washington, D.C., Feb. 5-7,1991.

[8] Department of Defense, *Department of Defense Trusted Computer System Evaluation Criteria (TCSEC)*, DOD 5200.28-STD, December 1985.

[9] National Computer Security Center, *Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria* (TNI), 31 July 1987.

[10] National Computer Security Center, *Trusted Database Interpretation of the Trusted Computer System Evaluation Criteria* (TDI), April 1991

[11] *Information Technology Security Evaluation Criteria(ITSEC)*, draft of 2 May 1991.

[12] Daniel Sterne proposed the terms automated security policy and organizational security policy in "On the Buzzword Security Policy", *Proceedings of the 1991 IEEE Computer Security Symposium on Research in Security and Privacy*, May 1991, Oakland, CA..

[13] Crawford, D.S. "Modelling Security Policy and Labelling Unclassified but Sensitive Information - A Canadian Perspective", *Proceedings of Standard Security Label for GOSIP An Invitational Workshop*, NISTIR 4614, June 1991.

[14] Biba, K.J., *Integrity Considerations for Secure Computer Systems*, MTR-3153, Rev. 1, Electronic Systems Division, Air Force Systems Command, United States Air Force, Hanscom Air Force Base, Bedford, MA, April 1977 (ESD-TR-76-372).

[15] D.D. Clark and D.R. Wilson, "A Comparison of Commercial and Military Computer Security Policies", *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, Oakland, CA, April 1987.

[16] William Wilson of Arca reports (San Antonio, Dec. 1991) that the SEVMS integrity portion is not well-utilized because of the absence of a standard integrity user clearance structure like the widely-implemented DOD user clearances for confidentiality.

[17] European Computer Manufacturers Association, *Security in Open Systems, A Security Framework*, ECMA TR/46, July 1988.

[18] Dobson, John and McDermid, John, "A Framework for Expressing Models of Security Policy", *Proceedings of the 1989 IEEE Computer Society Symposium on Security and Privacy*, May 1-3, 1989, Oakland, CA.

[19] Steme, Daniel, Martha Branstad, Brian Hubbard, Barabara Meyer, Dawn Wolcott, "An Analysis of Application-Specific Security Policies", *Proceedings of the 14th National Computer Security Conference*, October 1-4, 1991, Washington, D.C.

[20] Haigh, T., O'Brien, Fine, Endrizzi, Wood, and Yalamanchi, "Assured Service Concepts and Models", draft Final Technical Report, Contract Number F30602-90-C-0025, October 1991, CDRL A007, vol. 1 and 2.

[21] Burns, R.K., "Referential Secrecy", *Proceedings of the IEEE Computer Security Symposium*, Oakland, CA, 1990.

[22] Maimone, B. and R. Allen "Methods for Resolving the Security vs. Integrity Conflict", *Proceedings of the Fourth RADC Database Security Workshop*, Little Compton, R.I. April 1991.

[23] Feiler, P., "Experiences with Software Process Models, Session Summary: Policies", *Proceedings 5th International Software Process Workshop*, Kennebunkport, ME, October 10-13, 1989.

[24] Moffet, J. and Sloman S., "The Representation of Policies as System Objects", Domino Report:B1/IC/6.1, August 20 1991; "The Source of Authority for Commercial Access Control", IEEE Computer, February 1988.

[25] Sibley, E., Michael, J.B., and Richard Wexelblat, "An Approach to Formalizing Policy Management", CECCOIA2-*Proceedings of the 2nd International Conference on Economics and Artificial Intelligence*, Pergamon Press, Oxford, England, 1992.

[26] Grenier, Guy-L, Richard Holt, Mark Funkenhauser, "Policy VS Mechanism in the Secure Tunis Operating System", *Proceedings of the 1989 IEEE Computer Society Symposium on Security and Privacy*, May 1-3, 1989, Oakland, California.

[27] Page, John, Jody Heaney, Marc Adkins, Gary Dolsen, "Evaluation of Security Model Rule Bases", *Proceedings of the 12th National Computer Security Conference,* Baltimore, Maryland, 1989.

[28] Abrams, Marshall, Leonard LaPadula, Kenneth Eggers, Ingrid Olson, "A Generalized Framework for Access Control: An Informal Description", *Proceedings of the 13th National Computer Security Conference*, Washington, D.C., October 1990.

[29] Bell, D.E., and LaPadula, L.J., "Secure Computer System:Unified Exposition and Multics Interpretation", MTR-2997, the MITRE Corporation, July 1975.

[30] Brewer, Dr. David F.C. and Dr. Michael J. Nash, "The Chinese Wall Security Policy", *Proceedings of the 1989 IEEE Computer Security Symposium on Security and Privacy*, Oakland, CA, 1989.

[31] Bell, D. Elliott Bell, "Putting Policy Commonalities to Work", Proceedings of the 14th National Computer Security Conference, October 1-4, 1991.

[32] Amdahl Corporation, Multiple Domain Feature, General Information Manual, Amdahl MM001501001[1:10]6-89.

[33] Honeywell Inc. B-Level Design Specification for the LOCK Operating System, CDRL A009, Contract MDA 904-87-C-6011, June 1987.

[34] The diagram and description combine our visualization of metapolicies resolving policy conflicts[1] with Dr. Marshall Abrams' diagram of a proposed ISO conflict resolution process for access control policies (unpublished) using Dr. Leonard LaPadula's voting concept for rule-based systems. [35].

[35] LaPadula, Leonard J. "A Rule-Base Approach to Formal Modeling of a Trusted Computer System", M91-021, August 1991.

THE NEED FOR A MULTILEVEL SECURE (MLS) TRUSTED USER INTERFACE

Greg Factor
Steve Heffern
Doug Nelson
Jim Studt
Mary Yelton

GDSS/MLS Program
Digital Equipment Corporation
721 Emerson Rd.
St. Louis, MO 63141

Keywords: User Interface, Multilevel Security, FIMS

Point of Contact: Greg Factor (314)991-6235

THE NEED FOR A MULTILEVEL SECURE (MLS) TRUSTED USER INTERFACE

Abstract

Today the Department of Defense (DoD) relies upon software which would cost many billions of dollars to replace, yet many commands require the operational benefits of applied Multilevel Secure (MLS) technologies. Traditional secure engineering practices would require that the majority of these existing DoD systems be rewritten from scratch. The ideal approach is to take advantage of emerging Commercial Off the Shelf (COTS) MLS products (Operating Systems, Databases, Networks, Compartmented Mode Workstations (CMWs), etc.) by securely integrating them into a certifiable MLS System without requiring applications to be engineered in a trusted fashion. When this goal is achieved, the DoD will receive the benefits of applied MLS technology without having to incur the costs of developing and maintaining custom MLS systems that are too large to be certified. A critical missing MLS COTS product required to build a complete, certifiable MLS system is a Trusted User Interface. This paper explores Trusted User Interface requirements.

MLS Program Overview

The United States Transportation Command (USTRANSCOM) in conjunction with the Headquarters (HQ) Military Airlift Command (MAC) has established a DoD MLS command center testbed.[1] Two objectives of this program are to provide MAC with an MLS system which meets MAC's operational requirements, and to provide the DoD with a methodology of implementing MLS technology in HQ MAC and other command centers.

The objectives of the MLS Testbed are to:

- evaluate emerging COTS MLS products for capabilities both present and missing
- integrate COTS MLS Products into a single MLS System with a minimal amount of Trusted Computing Base (TCB) extensions
- utilize as many COTS products as possible, to reduce the amount of trusted code which must be developed

[1] The work reported in this paper was performed by Digital Equipment Corporation for the Military Airlift Command under contract F11623-89-D0007

- isolate COTS MLS products from the Trusted Computing Base, to allow new and emerging COTS MLS products with higher levels of trust to be incorporated into the system as they become available

- adhere to industry standards and the open systems philosophy whenever possible to facilitate lifecycle maintenance and the ability to swap out antiquated hardware and software for more modern products inexpensively

- define the standard requirements in building a MLS system and feed that information to both government organizations who are building MLS systems and product vendors who are building COTS MLS products.

Evaluation of available and emerging Commercial Off the Shelf (COTS) MLS products (Operating Systems, Databases, Networks, Compartmented Mode Workstations (CMWs) has identified a key missing COTS MLS product: a Trusted User Interface. The Trusted User Interface is the display portion of the overall system which receives the MLS data which is retrieved by way of the secure Operating System, trusted Database and secure Network.

One operational requirement for the USTRANSCOM/MAC MLS system is to be able to label different classifications of data on a single screen. The end users, in performing their job, are required to pass information about a particular missions or exercises over unclassified media (phones as an example). The users therefore need to be able to distinguish between the unclassified and the classified data, populating an individual screen, in a *trusted* manner. Without a Trusted User Interface, there is no way to display *trusted* security labels on the screen.

Since existing and emerging COTS MLS products are at the B1 security level, and many areas within the DoD have established a need for B3 and A1 systems, it becomes even more important to use standards that support open systems. Standards like POSIX, ANSI SQL, TCP/IP and FIMS. With the rigorous use of industry standards, emerging products with a higher level of trust can easily replace less mature MLS products. This allows MLS systems to migrate to a higher of level of trust without requiring applications to be rewritten to incorporate new COTS products.

The Form Interface Management System (FIMS) Standard

History

The Form Interface Management System (FIMS) is a proposed industry standard for forms processing that has been under development since the early 1980's and is being considered by the ANSI/ISO accreditation standards committee.

One of the key benefits of this proposed standard is the separation of the form processing from the specific application processing. The separation encouraged by the FIMS standard would result in more maintainable code and a less tightly coupling to display device peculiarities.

FIMS Components

In basic terms, the FIMS Standard is described consisting of three separate components:

- Independent Form Description Language (IFDL)
- Forms Control System (FCS)
- FCS application interface

Independent Form Description Language (IFDL)

The Independent Form Description Language (IFDL) describes an entity called a form.

Forms Control System (FCS)

The run-time component of FIMS is the Forms Control System (FCS), which controls the form's interaction with both the user (through a variety of input and display devices) and the application program.

FCS Application Interface

The interface to the FCS from the application is typically written in a higher level language (such as Ada or C). The interface performs all interaction with the user through the form by calling the FCS.

MLS User Interface Requirements

In defining the requirements that the typical user desires in the MLS system, we have come across several that point to the need for a Trusted User Interface to handle the security attributes of the form and the data. These attributes are not addressed by the proposed FIMS standard but should be incorporated into that standard.

Need for MLS Screens or Forms

One of the most basic needs is that the typical MLS system user would like to see, on a single screen, the highest level of data for which he or she is cleared. A TOP SECRET user, for example, wants forms or screens that potentially display sets of data that are composed of TOP SECRET, SECRET and UNCLASSIFIED data elements all on the same screen.

Need for Trusted Labels

In addition, the user requires that the data on the screen be labeled to clearly mark the data fields at the appropriate level of security. Furthermore, the labels need to be *trusted labels* and not advisory labels.

Need for a Labeling Standard

These trusted labels need to be associatable with their corresponding labels in the MLS operating system, MLS database and the MLS network products at a minimum.

Need for Security Attribute Processing Capabilities

The Trusted User Interface should also be able to extract from the expanded MLS data dictionary, associated with the MLS database, a variety of security attributes. These attributes should be handled by the Trusted User Interface and not by the application.

Attributes, similar to data validation, like a field being always at system low security level, i.e., always UNCLASSIFIED. When a user would try to enter Classified data into a field that is defined to be always Unclassified the Trusted User Interface should prevent this.

Or a data element having a security range, for instance UNCLASSIFIED through SECRET, but never TOP SECRET. If a user tries to enter data outside the defined security range, the Trusted User Interface should also prevent this.

Or relations between data elements, things like aggregation rules where when a specific data element is classified, other related UNCLASSIFED data elements immediately become classified and the database is updated to reflect the classification change. This should also be a capability of the Trusted User Interface.

Conclusion

- There is a need for a Trusted User Interface
- The Trusted User Interface should be FIMS compliant
- The additional security requirements are relatively easy to implement
- It should be possible to develop a B3 Trusted User Interface

References

Dan Frantz and Tom Poevey, *The Forms Interface Management System (FIMS): A Proposed Industry Standard*, August 1987

NETWORK SECURITY VIA DNSIX

Integration of DNSIX and CMW Technology

Howard A. Heller
Harris Corporation
M.S. W2/7742
P.O. Box 98000

Introduction

With the proliferation of computer networks in secure environments, it has become apparent that the security of network communications must be addressed. Currently, most computer security implementations are host based, that is they concentrate on security within a given workstation without regard for the communications between these workstations. This paper describes an internal research and development (IR&D) effort completed in early 1991 at Harris Corporation, Information Systems Division (ISD). The objective of the program was the integration of two leading edge security technologies to form a secure multilevel heterogeneous networking environment.

The IR&D team was to integrate the DODIIS Network Security for Information eXchange (DNSIX) protocol into a Compartmented Mode Workstation (CMW). The results of the study show that it is possible to integrate the two technologies to produce a fully functional Commercial Off-The-Shelf (COTS) DNSIX CMW capable of supporting a network operating in compartmented mode.

This paper will concentrate on three unique aspects of the IR&D; integration issues, design methodology, and leveraging the DNSIX IR&D into new business opportunities.

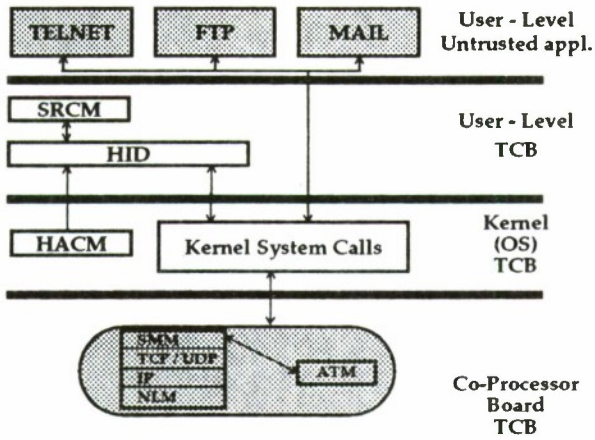
Background

The Department of Defense (DOD) Intelligence Information System (DODIIS) is a community of sites that exchange intelligence information via the inter-computing network known as the DODIIS Internet. The security requirements for the DODIIS Internet were formally defined by the Defense Intelligence Agency (DIA) in the 1986 publication entitled "DODIIS Network Security Architecture and DNSIX [1]".

What is DNSIX?

DNSIX is a protocol that satisfies the technical requirements for internetworking in DODIIS. This software supplements, and works in conjunction with, the standard DODIIS networking suite known as TCP/IP (Transmission Control Protocol [6]/Internet Protocol [7]).

DNSIX defines capabilities that manage and audit network sessions, maintain user security levels across the network, monitor access control to protected resources in the host systems, and enforce the network mandatory access control policy. These capabilities will be expanded on in the following paragraphs:



DNSIX Modules

DNSIX manages network sessions by establishing a special DNSIX connection with the remote host prior to any data exchange. A "DNSIX handshake" is used to pass information about the initiating host to the associated host including: host name, network level, application identifier, user name, terminal identifier and more. This information is sent to the associated host in the form of a Session Request message and is mediated upon by the associated host. This means that the information received by the associated host, along with its security policy rules, determine whether the network connection should continue or not. The DNSIX module which makes this security policy decision is called the Session Request Control Module (SRCM). The response is sent back to the initiating host via the Session Request Response message. The DNSIX module which handles this session management is appropriately called the Session Management Module (SMM), and resides on the intelligent ethernet board.

DNSIX audits network events such as new connections and closed connections. Security events are also audited. These consist of various levels of access exceptions. The DNSIX module which performs the audit function is called the Audit Trail Module (ATM), and the module which detects access exceptions is called the Host Access Control Module (HACM).

The definition of a DNSIX access exception is not clearly defined in the DNSIX specification (Our interpretation is explained later in this paper). The result of a DNSIX access exception is to abort the network session.

DNSIX maintains the user's security level across the network connection by passing the label to the associated host as part of the initial DNSIX handshake. If the session is accepted by the associated DNSIX host, the remote process started on behalf of the user is set to the security level passed in the DNSIX handshake.

The mandatory access control enforced by the DNSIX specification dictates that the security level on all incoming and outgoing datagrams will be checked to ensure the are within the allowable range for the particular host. If a datagram is created for transmission or received from the network which is out of the allowable range, the datagram is discarded. The DNSIX module which performs this function is called the Network Level Module (NLM).

The Host Interface to DNSIX (HID) module coordinates the passing of messages between the various other modules. All messages passed between the SMM and the Kernel are routed through the HID.

The figure above shows the various modules of DNSIX. The unshaded boxes are the modules implemented by Harris during the study.

The DNSIX protocol is documented in "DNSIX Detailed Design Specification [2]" and the "DNSIX Interface Specification [3]". Communication Machinery Corp. (CMC) was the vendor supplying the DNSIX/TCP-IP software and hardware support.

What is a CMW?

The Compartmented Mode Workstation (CMW) is a computer workstation whose operating system is designed to handle classified information with differing security compartments in a data fusion environment. The CMW is designed to meet the requirements specified by DIA/MITRE "Security Requirements for System High and Compartmented Mode Workstations" [4].

A CMW handles security for non-networked computers by labeling objects with security labels and allowing only authorized users the ability to manipulate those objects. All objects possess a classification and each user has a clearance. For example, in order to determine whether a person should be allowed to read a document, the person's clearance is compared to the document's classification.

Harris Corp. had previously developed a CMW prototype based on AT&T's System V Unix release 3.2 and X Windows release 11.3. This prototype served as the platform for our DNSIX integration activities.

Integration of Technologies

The DNSIX protocol works in conjunction with the TCP/IP protocol and, in CMC's implementation, executes on an intelligent ethernet board. The communication between the board and the host computer is accomplished using two distinct interfaces. The Berkeley Unix Socket mechanism has had some strategically placed kernel calls added to it to provide the operating system with information pertaining to the opening and closing of DNSIX sessions. Also, a special trusted mechanism was set up for passing DNSIX specific information between the board-side code and the kernel which could not be integrated into the socket mechanism.

The following events are communicated between the host and the board:

- A TCP open request has been made.
- A TCP close request has been made.
- The information obtained from the DNSIX handshake.
- Access exception messages.
- Acceptance / denial of session requests.
- Abort messages.

Socket Handling

The process of opening and closing a connection is handled by the socket mechanism. When an application opens a socket, the socket code calls a new routine in the kernel to retrieve all the relevant security information. This information is stored by the kernel in the process table. When the application calls the connect socket call, the socket code passes the security information to the board. The software on the board now begins the DNSIX handshake. If the remote workstation approves the session, the socket code is informed to allow the TCP connection to be initiated. If the session was denied, the TCP connection is refused. On successful connections, the DNSIX session closes when the associated socket is closed.

The other DNSIX events listed above also need to be passed between operating system and the board, but, unlike the events above, there are no corresponding socket calls for these DNSIX events (e.g. - open socket call corresponds to an open DNSIX message).

To provide the communications path between the kernel and the DNSIX code, a trusted application was created to act on behalf of the kernel. This trusted application is called the Host Interface to DNSIX (HID). The HID communicates

with the DNSIX code residing on the board by using a special trusted loopback mode provided by CMC within the TCP/IP software. The HID communicates with the kernel by using two new system calls (described later).

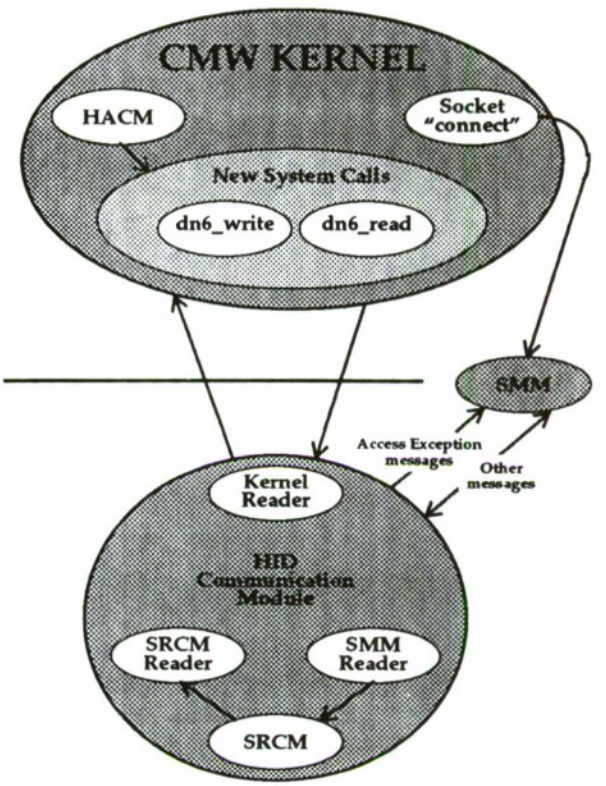
Upon startup, the HID opens two trusted connections to the board. One is for general DNSIX messages. The other is for Access Exception messages only. The board-side code will not go into operational mode until these connections have been established.

This provided a way for the HID to communicate with the board, but a mechanism was still needed to communicate with the operating system.

We added two new system calls to the kernel. One, called "dn6_read", provides the HID with a means to receive messages from the operating system. The second, called "dn6_write", provides a way of sending messages to the operating system. This provides a full path between the operating system and the board. Messages from the board are first sent to the HID via the TCP connection, and then sent to the operating system by means of the "dn6_write" system call. Messages from the operating system are sent to the HID via the "dn6_read" system call and then are routed to the board via the TCP connection.

HID Processing

The HID is basically a message switch. It receives messages from the board or the operating system and determines where to send them. Not all messages from the board are sent to the operating system. In particular, during the DNSIX handshake, a "Session Request" message is sent to the remote workstation. This is the message which contains all of the relevant security information about the initiator of the session. The DNSIX software on the board sends this message to the HID. The HID does not send this message to the kernel. Instead, it sends the message to a separate program called the Session Request Control Module (SRCM). The SRCM receives the message and uses its contents to determine whether to approve or deny the session. In any case, the SRCM sends its reply to the HID, which switches the message to the board. A discussion of the various other functions of the HID are beyond the scope of this document.



DNSIX Data Flow

Access Exceptions

Access exception messages are generated by the operating system itself. A CMW generates an internal access exception any time a user or process attempts to access an object to which it is not authorized. These exceptions are usually not malicious. For example, the files in a directory on a CMW are not necessarily at the same sensitivity levels. A user only sees those files which he has access to, and will not see that any

other files exist. This is called directory virtualization, and is a common CMW feature. When the user looks at a listing of a directory, the software (the 'ls' command) attempts to access all of the files in the directory. However, since the command runs at the same security level as the user, and the user's security level is not equal to, or higher than all of the files, an access exception occurs. This is a normal occurrence and in most cases the user is not even aware of the fact that it happened.

The DNSIX protocol provides a means of reporting access exceptions from the operating system to the board-side DNSIX software. It also specifies that any DNSIX access exception will cause the DNSIX session to be aborted. The specification does not, however, specify which host events should cause a DNSIX access exception to be sent. We decided to make this feature extremely flexible.

The kernel checks for access exceptions within each system call routine. These routines are used by programs to perform specific system functions. Many of these routines attempt to access an object on behalf of a user program. If the program is not authorized to access that object, an access exception is flagged.

Since the CMW doesn't abort a local users' session in the above scenario, we didn't feel that this should cause a DNSIX access exception. However, we decided to allow access exception processing to be configurable on a per-system call basis. Each system call has a flag associated with it which is used to determine if an operating system access exception should cause a DNSIX access exception. The default configuration has them all turned off. Any combination of system calls can be used, for instance, any access exception caused by opening a file could cause the DNSIX session to be aborted.

The other means of supporting the access exception mechanism is through another new system call, called "dn6_except". This was provided because of the uncertainty about what type of events would cause an access exception. With this system call, a trusted application can cause an access exception based on application specific security checks.

Access exception messages from the operating system are sent to the board via the HID as described above but with one difference. The TCP connection used to send access exception messages to the board is a separate dedicated connection so urgent security relevant messages will have processing priority.

Kernel Additions

The DNSIX interface code consisted mainly of additions and not many kernel modifications. The major structure added to the kernel was a table called the DNSIX Session Table. Its function is to keep DNSIX information about processes engaged in some stage of a DNSIX session.

The table is needed to create the link between the DNSIX Session Identifiers, which DNSIX uses to identify sessions, and the socket which the kernel uses to identify sessions. The table contains:

- Process Identifier
- Session Identifier
- Socket
- Device Number
- Session State
- Other Information

A table entry is:

- Created when a process calls the "socket" system call.
- Deleted when a DNSIX session is closed or aborted.
- Duplicated when a process calls the "fork" system call.

A DNSIX session can be in any of the following states:

- Idle: No session
- Init1: Socket created, no connection yet.
- Init2: Session request sent, no reply.
- Active: Session is active.
- Active_NonDN6: Session is active with a non-DNSIX host.
- Except: An access exception message has been sent.
- Other states are beyond the scope of this document.

All empty table entries have their state set to Idle. When a process calls the "socket" system call, the corresponding session state entry in the table changes from Idle to Init1. Within the "socket" system call, the DNSIX software retrieves the security information from the process table and sends this information to the board. At that point the session state changes to Init2. Once a reply is received from the remote system indicating that the DNSIX session was approved, the session state changes to active. The state remains active until either the session is closed or the session is aborted. When either of these events occur the session state returns to Idle.

DNSIX provides connectivity to non-DNSIX hosts as long as this is supported by the local security policy. If a connection is made to a non-DNSIX host, the session state gets set to Active_NonDN6 rather than Active.

If an access exception occurs while a session is active, the state is set to Except. This can only happen when the session is currently in the active state. When set, the state of the session is both Active and Except. This state is necessary in order to keep the process from continuing its activities while the connection is being aborted.

The only other kernel changes necessary were minor modifications to some system calls. In particular, in the fork system call, we added a DNSIX call which would duplicate the DNSIX session table entry of the parent process for the new child process. Unix performs this same procedure for files, since child processes inherit the resources of their parent processes.

In the kernel's exit and close routines, which are called when either a process is terminated or the connection is closed, we added a DNSIX call which would delete the process entry from the DNSIX session table. This keeps the table up-to-date when a process is terminated before closing its connection and when the connection is closed in the normal way.

The kernel's reference monitor is the routine which determines if there have been any access exception violations. We added a routine which determines if the exception was caused by a DNSIX session, and if so, if the DNSIX exception flag was set. If the flag was set, then the exception is reported to DNSIX which will abort the session.

IR&D's are real work!

The key to the success of the project resulted from creating a new design paradigm for the IR&D. Whereas most IR&D's are typically known for their lack of documentation and design, we decided to treat this one as a "real" program. This concept basically tailored the formal documentation used by Harris on "real" projects. Some of the documents not created for this project include:

- Software development plan
- Software quality assurance plan
- Software configuration management plan
- Software requirements specification
- Software test reports

We modified the standard software engineering methodology to meet the needs of the project. Our modified methodology consisted of:

1. Preliminary Design
2. Design Review
3. Prototype software
4. Detailed Design
5. Design Review
6. Implementation
7. Test

The reason for following this methodology was two-fold. First, we wanted to add some structure to the typical ad-hoc IR&D design process. Secondly, we wanted to use this program as a training ground for "real" program documentation. This was especially important for us because of the recent new hire assigned to this project.

The protocol software, which runs on an intelligent ethernet board, was being developed in parallel by CMC, a third-party vendor. Because our design was totally dependant on the interface to this software, we needed to keep in close contact with CMC so we could modify our design based on their changes.

By providing a Preliminary Design Document to the vendor, we were able to solve some major incompatibilities before any code was ever written. This worked both ways. Our design brought out many discrepancies in the vendor's software as well. The communication between the two engineering teams including documents, phone correspondence and electronic mail was a major benefit to the project.

We found that the decision to build a prototype before the detailed design was appropriate. Since the design seemed to be constantly changing we decided to freeze the design and prototype what we had. Since we didn't have the hardware or protocol software yet, we designed a screen oriented test program which could simulate the DNSIX messages passed back and forth between the host and the board. The test program would also receive the messages created by our prototype software destined for the board. The prototype software used extensive logging of debug messages and the test program would display these messages without having to exit the tester.

The outcome of the prototype was very significant. Many details were overlooked because they were based on assumptions about how Unix worked or because by actually doing the coding we had to think at a much more detailed level.

The prototype software ended up producing approximately 90% of the host-side software necessary and stabilized the design for both CMC's interface and board-side software and our host-side interface software. Because of the success of the prototype stage, we were able to create a Detailed Design Document based on the Preliminary design with modifications from the lessons learned during prototyping.

The implementation phase began when we invited the CMC engineer to Harris. We made plans for him to stay for 2 weeks, on the assumption that there would be many problems integrating the two pieces of software.

The first day was spent setting up equipment and installing software. The second day surprised us all. After loading the CMC code on the ethernet boards of two machines containing our new software, we attempted to open a DNSIX connection. It worked! For the first time, DNSIX 2.0 had been integrated into a CMW and a secure networking connection was established.

Of course everything didn't work perfectly. The rest of the two week visit

was spent trying different tests, debugging problems and making design changes on the fly. By the time the CMC engineer left, we had a very high level of operational integrity and were confident that our design was a good one.

We decided up front that we were not going to do a formal test plan. This was because the goal of the IR&D was to get as much experience as possible and to get as far along in the integration phase as time permitted. The testing was done on a per-feature basis, determining the correctness as features were added. These features were retested at the conclusion of the program. The goal of the IR&D was to make the final system have as much functionality as possible in order to make a convincing demonstration to potential customers. We feel the extra time spent on adding more functionality at the expense of additional testing was well worth it.

We feel the project went smoothly because of the following factors:

- Upfront design, but not overly detailed.
- Prototyping the design early in the project.
- Constant communication with the people that count (direct contact between the CMC and Harris engineers).
- Testing distinct modules separately and thoroughly before integration.
- Integration of modules completed with representatives from each team to facilitate quick turnaround time on code changes and quick answers on integration issues.
-

Win-Win Partnership

The partnership between Harris and CMC was a win-win situation. CMC, developing the DNSIX protocol software, was very limited in their testing because DNSIX was designed to be integrated into a secure workstation. DNSIX is tightly coupled to the operating system in order to extract the user's security labels and be aware of access violations. Because CMC lacked a secure workstation they wrote command line test programs which would simulate the actions of the secure operating system. This way of testing is awkward at best and makes many assumptions as to how the operating system may perform its tasks.

Harris, seeing the potential success of the CMW, determined that the obvious next step to secure computing was the addition of secure networking. This would provide a trusted communications mechanism between a network of secure workstations. The DNSIX IR&D was important to Harris in that it provided us with the necessary knowledge and experience to present leading-edge solutions to our customers security problems.

Teaming with Harris on the integration of the two technologies provided CMC with the opportunity to exercise their technology in an operational environment. This reduced their engineering time and decreased the risk of software errors.

Leveraging DNSIX Technology

By being the first to integrate the DNSIX protocol with a CMW, Harris has proven its commitment to provide leading-edge security technologies to its customers. The engineers in our Computer Security Group (Harris-ISD) used this activity to develop network security expertise for ongoing programs that will eventually require DNSIX-like capabilities.

We currently have a DNSIX/CMW demonstration capability in our Computer Security lab and have received an enthusiastic response from customers who have seen its capabilities. The ability to demonstrate a genuine working system produces much more excitement about a

technology than white papers or visual presentations.

The knowledge gained from our DNSIX efforts will be directly applicable to the integration of other security protocols into customer environments. In particular, the SDNS SP3 and SP4 security protocols, which provide security for levels 3 and 4 respectively of the OSI Reference Model [5], are of interest to our customer communities. We have also observed a need for security at the application level such as secure electronic mail and secure multimedia. These are applications which usually contain correspondence between two or more individuals and are most likely to contain sensitive material.

The DNSIX IR&D has given Harris many benefits and no drawbacks. We feel this is because the project focused on developing a demonstratable system rather than a pile of documentation describing the technology. Our tailored design methodology and early prototyping played a major role in the success of the project. By being able to show our customers a technology at work, they no longer wonder if the technology is possible. Instead, the customer begins to point out the potential the technology could have in their environment and the various uses they have in mind for it.

References:

[1] Defense Intelligence Agency, May 1986, DODIIS Network Security Architecture and DNSIX, DRS-2600-5466-86, DODIIS System Engineering Office, Arlington, VA.

[2] Defense Intelligence Agency, April 1990, DNSIX Detailed Design Specification, DDS-2600-5985-90, DODIIS System Engineering Office, Arlington, VA

[3] Defense Intelligence Agency, April 1990, DNSIX Interface Specification, DDS-2600-5984-90, DODIIS System Engineering Office, Arlington, VA.

[4] Defense Intelligence Agency, Nov 1987, Security Requirements for System High and Compartmented Mode Workstations, DRS-2600-5502-87, DODIIS System Engineering Office, Arlington, VA.

[5] ISO 7498 Basic Reference Model for Open Systems Interconnection

[6] MIL-STD-1778, August 1983, Transmission Control Protocol

[7] MIL-STD-1777, August 1983, "Internet Protocol"

NEW DIMENSIONS IN DATA SECURITY

The innovative DES-chip called *SuperCrypt* allows for development of secure computer systems without the current limitations inherent in most chips currently available.

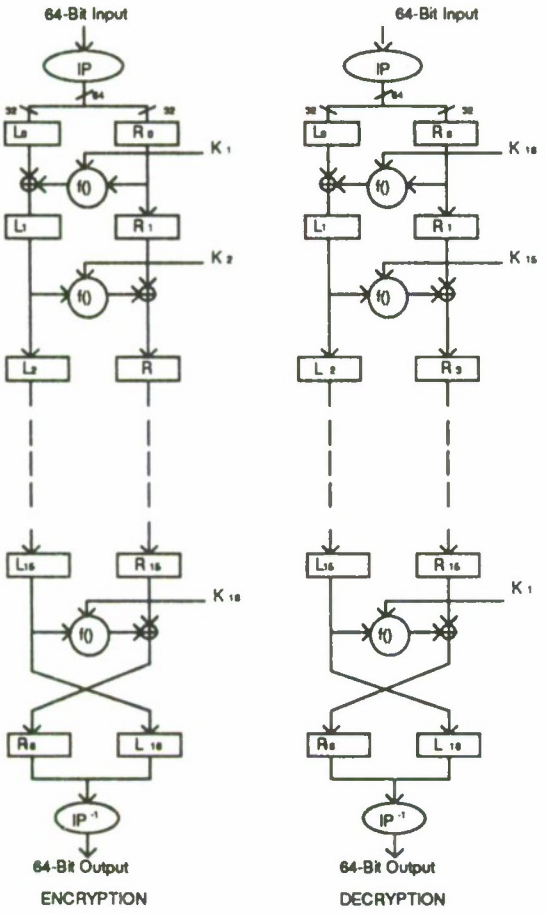
Karl Heinz Mundt
CE Infosys
512-A Herndon Parkway
Herndon, VA 22070
703-435-3800

Despite ever increasing importance, achieving reasonable data security in the fields of computer based information processing and telecommunications, has proven to be implemented only with the greatest of efforts. One reason is the rapid developments in information systems, such as networks, which are increasingly dependent on growing interoperability. Also, the availability of distributed computing power is steadily increasing. The incredible growth of data being archived on mass storage devices, distributed in LANs, and telecommunicated over the public telephone network has created a rising percentage of security sensitive data. Needless to say, it is in the best interest of the system users to protect this data from unauthorized tampering.

Providing *absolute secure physical access control* of sensitive data, especially in multi-user environments or freely accessible channels such as the telephone network is practically impossible. Providing a *logical access control using data encryption* has proven to be far more effective. The science of cryptography provides one of the most effective solutions.

Data Encryption Standard

Probably the best known and the first standardized encryption method is the Data Encryption Standard (DES), which was standardized and published in 1977 by the U.S. National Bureau of Standards. DES is a symmetrical block cipher algorithm characterized by the same key being used for en- and decryption of a message. The coding operation processes plain-text blocks of a fixed length; in this case 64 bit. The data encryption standard essentially consists of 16 iterations of permutations and substitutions being applied to the plain text for the encoding, or in reverse order to the ciphered text for the decoding operation. After an initial permutation (IP) the input data is sequentially routed through 16 blocks of XOR and specific DES substitutions, dependent on the bit pattern, with data from a function table. Finally a reversed input permutation IP^{-1} is performed.



Picture 1: Structure of DES Algorithm

The most important initial parameter of DES is a key code of 56-bits length, from which 16 partial keys are generated. Generating these 48-bit keys is part of the DES-algorithm and is achieved by permutation and a 16-level shift-function. The difference between encrypting information and decrypting is the way in which keys are generated. To decrypt the keys are presented in inverted order. The number of possible encryption results is limited only by the length of the 56-bit key used and is consequently 2^{56} (about $7*10^{16}$).

Different DES-Modes

Various modes of operation are standardized for DES to drastically decrease the chance of attacking the encrypted data by statistical methods. Long message streams; however, need additional protection by transforming identical plain text into non-identical encrypted counterparts. Some of these schemes are known as Cipher-Block-Chaining (CBC), Output-Feedback-Mode (OFB) and Cipher-Feedback-Mode (CFB). DES not only protects encrypted data against unauthorized access, it also reveals any manipulation to the encrypted data, i.e. due to transmission errors. This application of DES has been standardized in 1986 as Message-Authentication-Code (MAC) at ANSI. The application of MAC is a valuable feature for example in the area of electronic banking.

Problems Inherent in Applying Cipher-Methods

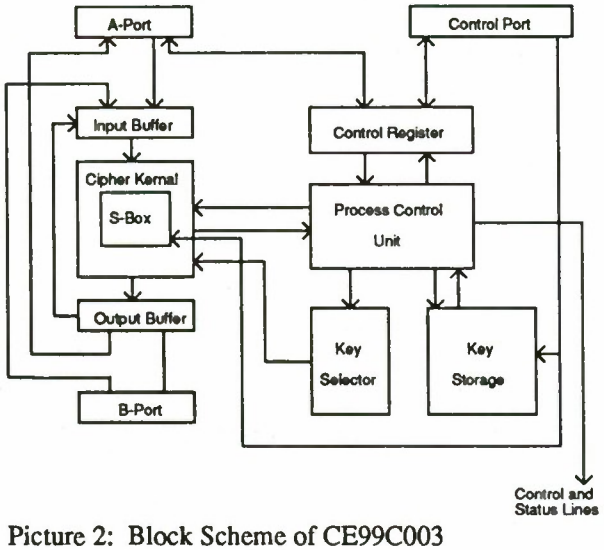
A number of products using the above mentioned cipher-methods are already certified by American institutes and are predominantly used in banking applications. A major drawback of secure cipher-methods is that they usually have a negative influence on the computing speed of the systems in which they are used. The known software and hardware implementations of DES have not been able to keep up with the requirements of current systems in respect to their encrypting and decrypting speed. This is especially true for software implementations of DES because of the problems inherent in programming bit-permutations, and results in time-consuming 64-bit encryption operations (which even on modern microprocessors need several milliseconds). Previous hardware implementations achieve maximum

encryption speeds of 40 Megabits per second.

The *SuperCrypt* Chip

SuperCrypt CE99C003, a highly integrated ASIC developed by CE Infosys, has a number of advantages:

- high encryption speed (160 Megabit per seconds at 33 MHz chip clock),
- loadable algorithms, and
- built in key management functions.



Picture 2: Block Scheme of CE99C003

SuperCrypt Architecture

SuperCrypt architecture realizes a physical division of security relevant functions such as the loading of keys and algorithms and the use of the data paths. For this purpose, *SuperCrypt* uses a data port A and a control port C. The 8-Bit control-port C is used when downloading the S-box contents or the keys, and when the access rights for data port A are defined. The 32-bit wide A-port, which also supports 8- and 16-bit accesses and automatically performs a bus-conversion, is used for fast data transfer. Internal registers are used for initializing *SuperCrypt* and can be accessed both from the A-port and the C-port directly using address lines, and indirectly using pointer-registers.
Data port, B, is used for transferring 8-, 16- and 32-bit data and allows for direct-buffer applications. In direct-buffer mode, data is fed into *SuperCrypt* A-port and read out to the B-port after en-or decryption directly into a buffer (RAM). Data can also be read from a buffer into the B-port and then made available at the A-port. This ensures optimal data-throughput even if slow bus-systems are used.

Special Features

SuperCrypt supports the DES-algorithm and all operating modes discussed above with maximum data-throughput because of an internal feedback-path. Even more complex variations of DES are supported by:

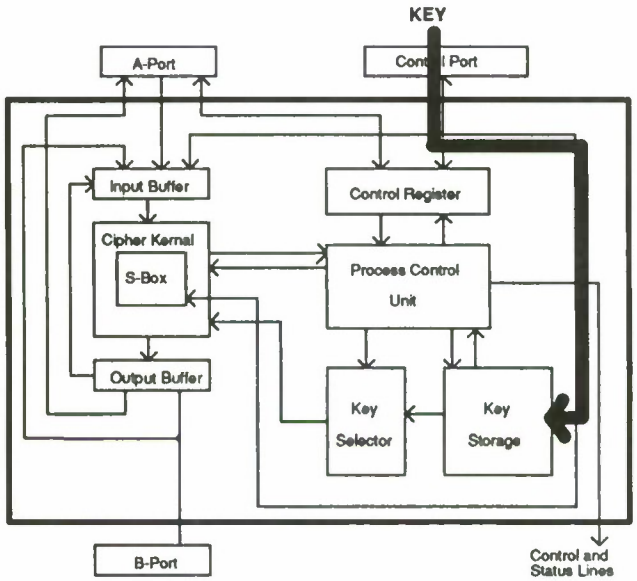
- **increasing the effective key length to 112 bit**
- **or by changing the contents of the S-boxes.**

***SuperCrypt* offers loadable substitution boxes (S-boxes).** The most significant cryptographic component of DES is the S-boxes contents. The methodology of the S-box construction can have far reaching implications for the security of the algorithm. The major nonlinear component of DES is the function *f* of the S-boxes. This nonlinearity gives DES its significant cryptographic strength. The function *f* takes as input 32 bits of partially enciphered data and 48 bits of key and produces 32 bits of partially enciphered data as output. *SuperCrypt's* loadable S-boxes provide the opportunity to create proprietary algorithms or load newer algorithms of greater strength.

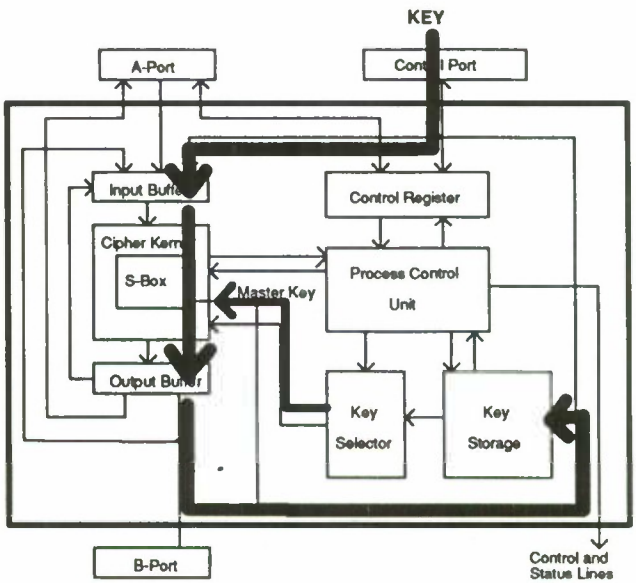
The following features are unique to *SuperCrypt*:

- **the internal key-storage of up to 16 keys**
- **and the Master-Session-Key concept,**

both of which greatly simplifies the handling of external keys. This is a potential point of attack in other systems not supporting this feature. The Master-Session-Key concept allows the often necessary public transport of encrypted keys, called Session Keys, which are decrypted internally and consequently stored with a Master-Session-key only when loaded into *SuperCrypt*. The Master-Session-Key feature is an optional method to load keys into the key storage. Normally the keys are stored in its original pattern in the key storage. This optional feature encrypts or decrypts the key pattern you load and then stores it. This means that while loading a new key, a "session" key can be generated by encrypting it with a "master" key. The new "session" key is then stored in the key storage. This guarantees protection against tampering since the "session" key is computed only inside the chip. This feature allows distributing keys in a non secure area.



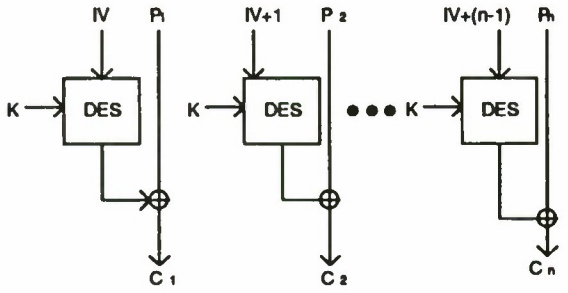
Picture 3. Standard key load procedure



Picture 4. Generating a session key with a specified master key

Another mode is the Key-Stream-Generator mode (KSG), a variation of the OFB-mode. Instead of using the cipher text of the previous block, an incrementing 64-bit counter generates

the initial vector of the cipher-function. This mode makes the realization of a pseudo-random generator possible and additionally allows random access to plain-text data within encrypted messages generated by this stream-oriented mode.



Picture 5: KSG Mode

A *Triple-Cipher* option is available with all operating modes. Plain-text data is DES-encrypted three times by first encrypting, then decrypting and encrypting once again with different keys each time. The triple encryption is automatically performed internally and greatly enhances the cryptographic strength of DES.

The encryption speed is linearly dependent on the speed of *SuperCrypt*. Encrypting a 64-bit block, excepting the Triple-Cipher option, takes 12 cycles. At the frequency rate of 33 MHz, this is equivalent to 360 ns or 160 MBit/s.

Specific Advantages

The advantages of an internal key storage are:

- a high degree of protection against attacks
- and the exceptionally high speed with which keys are changed.

After loading the keys they are no longer accessible from outside the chip. Using the integrated key-cipher to realize a Master-Session-Key concept makes it possible to perform ciphering with keys only decrypted after loading into *SuperCrypt*. This provides a protection feature against cryptographic attacks. Apart from selecting the required cipher mode only the key to be used for cipher must be addressed upon initializing. Different write attributes can be attached to each key, which can govern the usage of the key on data or the use for en- or decryption with a particular key. CMOS technology manufacturing makes buffering the key-storage or the S-Boxes possible.

To allow easy integration into Bus- or micro controller-systems, *SuperCrypt* supports Interrupt- and DMA-functions. Two DMA-channels are available. Two data-request signals make it possible to signal the readiness to accept or send data to a DMA-controller.

Adjustment and control of the operating modes is handled by programmable internal registers. To minimize the hardware address space, some of the register sets are indirect, that is, they can be accessed using a pointer-register. Access on the remaining registers is handled by setting an address pointer. A pointer that automatically increments after each register access, further simplifies initializing the register set. A programmer can load data for the indirectly addressable registers in one burst from a table. The C-port register set is used to control the security relevant functions such as loading keys and S-box contents, and therefore can only be accessed from the C-port. It also uses directly and indirectly addressable registers.

Initializing *SuperCrypt* mainly depends on the operating mode and the application requirements -- it may only require a few programming statements. If *SuperCrypt* is used in a battery-buffered application, initializing the key-storage and loading the S-boxes needs to be performed only once. A key may be loaded into any position of the key memory addressable by itself. Any key may be overwritten purposely later. Each key can be given three attributes upon loading and are stored with the key. These attributes can prevent a key from being used for a specific de- or encryption operation and protect a key from being overwritten. When loading a key, the Master-Session-Key function can be utilized. A key loaded encrypted is decrypted with a Master-Key stored at any address during the download, and is then stored as a Session-key. The key memory cannot be read. The individual keys are made accessible after downloading by initializing the Key-Enable register KYE.

The S-boxes contain 512 Byte and are loaded in one burst using the C-port data path register. Immediately after loading the S-box, memory may be verified. Further attempts to read are prevented by the chip logic and are only possible after a completely new download. If DES encryption is desired, the substitution data must conform to the standard. Nevertheless, a customer may load proprietary S-Box functions.

Application Support

The widespread use of information processing systems makes a number of encryption applications possible where data security is required.

For the first time available anywhere, *SuperCrypt* provides a hardware platform usable in real-time encryption without compromising the performance of the host-system.

To ease the design of new developments containing *SuperCrypt*, a design kit is available from CE Infosys. A fully functional 16-bit AT-adapter with numerous test-points that support all operating modes, detailed schematics, PAL-equations, and demonstration software in source and object code is included.

New Dimensions in PC Data Security

The overwhelming success of personal computers over mini and mainframe computers in the last decade can be primarily contributed to the strategy of providing an "open system" philosophy for both the hard and software. **Based on the *SuperCrypt* chip CE Infosys provides hard and software platforms with an "open architecture" for developing applications which require security functions.**

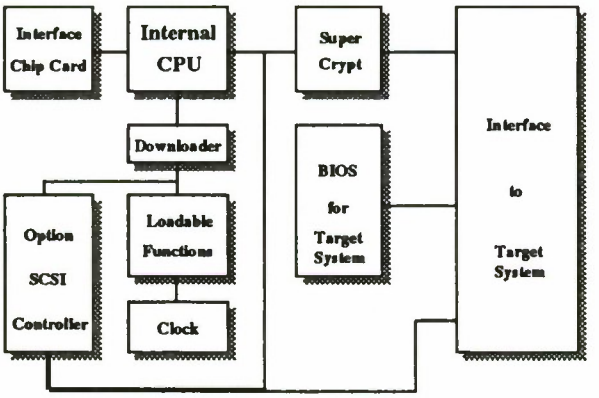
Software developers at the OEM, VAD, VAR, or End users level can create new or modify existing applications to fit virtually any security requirement without having to worry about timely and costly hardware and operating system software development. Complex security concepts and individual customization for the application environment can be implemented in just a fraction of the normal development time.

CE Infosys´ application examples includes numerous "off the shelf" functions ranging from basic en- and decryption to complete PC security functions including access, resource control and en- and decryption of files and/or any mass storage device or backup device.

"Off the Shelf" Functionality

Three basic hardware platforms are currently available all of which use the *SuperCrypt* encryption chip for the cryptographic functions.

The security controllers (SC8810, SC8820) are available for AT-Bus (ISA-Bus), Micro Channel Bus or PCMCIA Bus for laptops. A special version for ISA-Bus with an integrated chip card reader is also available for laptops.



Picture 6: Security Platform Architecture

The security SCSI controllers (SC5430, SC5440) are available for AT-Bus (ISA-Bus) and Micro Channel Bus and combine the functionality of the basic security computer with full SCSI device and operating system support.

Application Example 1 - MiniCrypt An encryption adapter using *SuperCrypt*

The cryptographic adapter, MiniCrypt, is available for AT-Bus (ISA-Bus) and Micro Channel Bus. It is designed to provide fast "file-by-file" encryption. MiniCrypt provides an extremely powerful platform by supporting most of ***SuperCrypt's*** functions independent of the operating system used. Most importantly, it supports ***SuperCrypt's*** high encryption speed capability. The half-length ISA and MCA adapters are a cost effective solution when existing software applications need to be speeded up by factors of 10-50 percent. Another option, the 64KB buffer, together with direct buffer mode operations, can double the encryption throughput. The product, "off the shelf", includes the adapter as well as DOS utilities for file encryption or decryption using DES. The sample application software is designed to meet a wide end user community and therefore, the loadable S-boxes invoke the Data Encryption Standard (DES). The API available for MiniCrypt allows one to create customized software that incorporate the enhanced features of triple DES or double length keys. For the end user desiring a proprietary algorithm, non-DES S-box contents and double length keys can be integrated to provide for very interesting cryptographic algorithmic strength. Encryption speed is limited only by the PC's internal bus system (2-2.5 MByte/s on an 8 MHz PC). MiniCrypt´s speed and its exciting cryptological functions open up whole new applications that take advantage of encryption, where previous software solutions either were too slow or used

weak algorithms to provide the necessary speed.

Application Example 2 - SC8810 A security adapter using *SuperCrypt* providing encryption, access control, and audit trail functionality.

Whereas MiniCrypt provides for file-by-file encryption, the SC88XX provides for a total PC security solution. The SC88XX hardware platform is available for:

- AT-Bus (ISA-Bus)
- Micro Channel Bus
- PCMCIA Bus for laptops
- A special version for ISA-Bus with an integrated Chip card reader is also available for laptops.

This high-end security platform provides "off the shelf" functions including:

- access control with optional chip card or smart card support,
- System resource control (floppy, hard disk, interfaces),
- Boot stop,
- "On-the-fly" permanent encryption of mass storage,
- User dependent encryption of floppy disk drives, or
- Selective user dependent file encryption.

As with MiniCrypt, one of the most important features that SC8810 introduces is "transparent" encryption for the end user. Security controllers that detrimentally effect personal computer performance are usually met with disdain from end users and are attempted to be circumvented. These applications designed with ***SuperCrypt*** avoid this serious end user concern. The Master-Session-Key management logic provides a standardized method of distributing keys, supporting security adapters in a corporate environment.

Previous version of SC8810 that used other than ***SuperCrypt*** encryption chips, required a metal cover to prevent tampering. ***SuperCrypt***'s Master-Session Key concept and internal storage of 16 keys make this not necessary. Sophisticated user profiles governing access to programs and system resources are supported, as are time profiles for regulating system access. Designed to meet a wide end user community, the loadable S-boxes invoke the Data Encryption Standard (DES). As in MiniCrypt, SC8810 can be customized to take advantage of triple DES or double length keys. Once again, for the end user desiring a proprietary algorithm, non-DES S-box contents and double length keys provide for very interesting cryptographic algorithmic strength. Both DOS and OS/2 operating systems are supported.

Application Example 3 - SC5430 A SCSI controller using *SuperCrypt* providing encryption of up to seven SCSI devices.

The Security SCSI Controller Platforms are available for AT-Bus (ISA-Bus) and Micro Channel Bus. They combine the security features of encryption and access control with the functionality of a high-end caching SCSI controller having full SCSI device and operating system support. Currently supported under DOS are:

- Hard disks
- Removal disks
- Magneto-optical disks
- Tape
- DAT
- CD-ROM
- and WORM devices

Novell and OS/2 drivers are under development. Each device, SCSI ID

dependent, can be permanently encrypted "on-the-fly" thus providing secure storage of sensitive data or programs. The processing speed of the SC5430 is equivalent to other commercially available SCSI controllers without encryption. Up to seven different peripherals can be attached, operated and encrypted/decrypted with different keys simultaneously at speeds fully transparent to the user. Secure tape backups on a network or "securing" data or programs on removable media are just two of the most common application areas.

Access control is done on a SCSI ID basis, utilizing chipcards. Each chip card maintains a table of:

- SCSI IDs
- whether the SCSI ID is encrypted or non-encrypted
- Keys associated with each SCSI ID

Removable media support (Iomega Bernoulli boxes, Syquest drives, Magneto-optical drives, tape drives, etc.) is handled with great sophistication. Multiple keys can be used at one SCSI ID. Each data cartridge, (tape, MO, etc.) can have a key assigned uniquely to it. The Master Session Key concept allows keys to be encrypted on the chip cards.

In networks a combination of security SCSI controllers for the server(s), SC881Xs for the workstations, and MiniCrypts for those workstations requiring only cryptographic functions but not access control can provide the solution for a "secure" network environment.

API Availability

A fully documented API (Application Programming Interface) as well as the development tool kits (C-Libraries, C-Source, BIOS-Routines, etc.) are available to provide numerous security functions.

API Security Functions:

- Access Control
- Boot Stop
- Chipcard Services
- Resource Control
- Cipher Engine (DES and other Algorithms)
- Cryptographic Tool Kit (one-way, hash, signature, etc.)
- Secure Key Storage
- Key Management
- Audit Trail (Logbook)
- Independent Time and Date

The API can be enhanced further to provide for future requirements or new algorithms. Software Development Kits (SDKs), for the application programmer and Product Development Kits (PDKs) which include the actual hardware design and manufacturing documentation are available.

Future Applications

SuperCrypt's flexibility combined with CE Infosys' Security Platform Hard- and Software open up whole new applications that can take advantage of encryption for security.

- At *SuperCrypt*'s speed makes possible encryption in "real time" of digital TV, one of the fundamental requirements for easy implementation of "Pay TV".
- Computers of the future can easilly have their disk, LAN adapters, HOST adapters or SCSI controllers equipped with *SuperCrypt* to cipher data "on-the-fly" as it streams through the controller. Backup tapes, disks, MOs and even CD-

ROM can store data in encrypted form and if grouped in blocks, each block can be ciphered with a different key.

- PC motherboards, Minicomputer and mainframe processor boards
- FDDI fiber optic networks for data communications and telecommunications
- Customers purchasing data on disk or CD-ROM would purchase those keys required to unlock the blocks of data actually purchased. This concept may very well revolutionize the methodology of software distribution or updating.

A Note on Compartmented Mode: To B2 or not B2?

Theodore M.P. Lee

Trusted Information Systems, Inc.
P.O. Box 1718
Minnetonka, MN 55345

Abstract

This paper calls into question current government computer security policy. That policy, as seen in DCID 1/16 and DoDD 5200.28, permits a B1 automated information system to be used in compartmented mode. In compartmented mode some users of a system are not formally approved for access to all of the information in it — even though all users have a uniformly high national security clearance — so as to minimize the damage caused by espionage. This note compares the reasons why compartmentation is used in the intelligence community with the ability of C2, B1, and B2 systems to resist various kinds of threats. That comparison convincingly demonstrates that at least a B2 system must be used in compartmented mode unless most of the benefits of having compartmentation are not to be sacrificed when an automated information system is used to handle and process compartmented information

Keywords

compartmented mode, DCID 1/16, trusted systems, need-to-know, threats, vulnerabilities, risks

Introduction

A system is running in *compartmented mode* if all users have national clearances at least to the level of all information in the system, the system has information from one or more compartments (that term is defined more sharply for our purposes later), and at least one user has not been approved for access to *all* compartments on the system. For the purposes of this note we are assuming everyone has at least a SCI (Sensitive Compartmented Information) clearance in accord with DCID (Director of Central Intelligence Directive) 1/14 [2]. It would appear that a system evaluated to at least the B1 level is necessary to support compartmented mode operation[1], since B1

[1]Strictly speaking, B1 is not logically "necessary." In a C2 system compartmentation in a benign environment can be (and is, albeit at risk) enforced analogously to the way it is in a strictly paper world. Users bear the responsibility of ensuring that each electronic document is marked with or in some fashion associated with its set of compartments. When an access list is created for a document (or some other form of access control that meets C2 requirements is employed) the user who sets or changes the access list has the responsibility to ensure that only people who are authorized access to all of the compartments in a document are placed on its access list or otherwise granted access to it (e.g., by giving only them a password for the document on systems that employ such means of access control.) The primary difficulties with this scheme (apart from its penetrability) are that a person has to keep track of the access approvals of, in principle, all other users and that it makes it difficult to use group accesses (if, say, all users in office A do not have approval for compartment B, I can't put "office-A" on a group access list for document marked with compartment B). Level B1 is the first level that can keep track of user access approvals and the

is the first level that supports the security markings needed to enforce compartmentation. DCID 1/16 [1] says that B1 is sufficient, mentions B2, but gives little guidance on when B2 might be necessary. DCID 1/16 is not alone in permitting B1 to be used in what amounts to compartmented mode: the NCSC "yellow books" [3], DoDD 5200.28 [5], and the DIS Industrial Security Manual (ISM) [6] all also permit B1 to be used when not all users have formal access approval for all information in the system, even though they all have a national clearance at least as high as the most highly classified information in the system. Anderson's paper [4] is one of the first public attempts to correlate trust levels with risk environments; unfortunately it is ambiguous in its treatment of compartmented mode: although it has "SCI" as an element in its clearance/classification matrix, it never uses the term "compartmented mode" and explicitly says for threat/risk category 1 (for which B1 is applicable) "there is no threat or risk since all users are cleared/*approved* for all material." (emphasis added) Each of these references attempts to define when a level of trust higher than B1 might be needed, but, especially in the case of the ISM, are not consistent with each other. In any case, [3], [4] ("with more than one category of SCI present, raise the threat/risk category by 1"), [5], and [6] condition the transition from B1 to B2 or B3 by the number of compartments for which not all users have access approvals; why the number of compartments is used rather than some notion of sensitivity of the actual compartments involved has always eluded me[2]. The remainder of this note is an attempt based on first principles to analyze further which level should really be used.

What is a Compartment?

First, we need to make clear what is meant here by "formal compartmentation" or "formal compartment", as that term is used in the intelligence community. For someone to have access to a formal compartment at least four events must happen: the person must have a "favorable" DCID 1/14 background investigation (BI)[3], there must be at least an administrative decision that the access is necessary and appropriate, the person must be given some kind of briefing on why the compartment needs (special?) protection, and he must sign a non-disclosure agreement specific to that compartment. The key point is that enforcement of compartment access is just as mandatory as enforcement of the national classifications: a person who has been granted access to information in a compartment does not have the right to bypass the administrative process and determine that someone else should see that information who has not been given formal access approval[4]. We

---

compartment markings on a document and use that as a basis for granting or denying access. (In a B1 system I *could* put "office-A" on the access list for the above B-compartment document and still be ostensibly assured that those in office A who did not have B access approval could not have direct access to it.)

[2]Roger Schell's explanation that "in most cases that gives you the right answer" is not intellectually satisfying, although I do understand how when one is thinking in the context of the common national control systems his reasoning makes some kind of sense.

[3]Since it appears that almost everyone in the intelligence community has at least that, the issue of national clearance is often ignored in discussions regarding compartments within the community. Also, since it is the case that the "national classification" of SCI information is in some sense a meaningless concept, the fact that SCI documents may bear different national classifications can usually be ignored in determining whether a system is running in compartmented mode or not. Even though a document might be marked SECRET WHIZBANG, where WHIZBANG is an SCI compartment, neither a SECRET nor routine TOP SECRET clearance is enough to permit access: one must have an SCI clearance which in itself gives one a TS clearance; the marking of SECRET says something about the sensitivity of the document (although I'm not clear what) but says nothing about what level of personal trust is needed to permit access to it since that is superseded by the (implied) SCI marking.

[4]We are here deliberately not addressing the fact that some, perhaps even a large number, of intelligence community officers and employees are authorized to "declassify" compartmented information (decide that for some reason it no longer needs protection) or to release it in whole or in part (usually in sanitized form) on a case-by-case-basis to people who do not have

could fine-tune that definition by saying something about who can establish compartments and what minimum kinds of administrative procedures (bookkeeping at least) must be involved, but this is close enough. In any case, different organizations, departments, and offices have slightly different procedures for creating compartments (as we define them here, whatever else they may call them), for deciding exactly what a person granted access is to be told and agrees to, who has to approve the creation of the compartment and what other administrative information and rules go along with it.

The question of whether C2 or B1 is "good enough" to enforce compartmentation then hinges on what the purpose of having compartments is and whether a C2 or B1 system sufficiently ensures that that purpose will not be thwarted.

Why is Compartmentation Used?
(or, Why Create a New Compartment?)

"Need-to-Know"

The first reason for having compartmentation is the "traditional" reason for need-to-know: to reduce the number of people routinely exposed to a given body of information. Three related purposes for this cautious approach come to mind:

Damage Limitation: to reduce the damage done by the bad guy who has slipped through the personnel security net, either as a mole or as someone who has been turned or duped[5].

Temptation: to reduce the chances that an ordinary cleared person (loyal, average, subject to the vicissitudes of normal life) will be tempted to abuse (inadvertently release, consciously try to sell or trade for gain, advantage, or to make a point) sensitive information which he just happens to have access to — the more classified information one sees, the greater the likelihood of running across something that proves just too tempting.

Attractiveness: to reduce the attractiveness of any single individual as an intelligence target — the less information a person has ready access to, the less worthwhile it is to take the risk of attempting to turn or exploit that person.

Other Purposes of Compartmentation

The above three reasons seem the same for ordinary need-to-know ("Can I see it?" "Yes, I like the color of your eyes and you seem to know something about this problem, so, go ahead") and compartmentation. (Remember that in most organizational security policies "need-to-know" is

---

formal access approval for it. To an outsider this may make the "mandatoriness" of a compartment label seem less "mandatory," but it isn't: the President can release any secret he likes to anyone he wants. The general rule still is that if I have possession of compartmented information I can only give it to someone else who is approved for all of its compartments; the rule in practice is just more complicated: in the national interest there are some people who are authorized to ignore the general rule on a case-by-case basis.

[5]Since we are for the most part not talking about field or tactical situations we are mostly ignoring the physical capture or over-run problem. Situations where those risks are present provide especially strong reasons for wanting damage limitation through compartmentation: capturing an agent can't roll-up the entire network. Note that over-run does have a direct relation to computers (what information can a person at a captured site be induced to continue to access) as well as a good indirect analogy (if the person is metaphorically over-run by a malicious program, what can it access using his authority?)

supposed to be exercised over ordinary non-compartmented classified information: a possessor of classified information is supposed to release it to someone else, no matter what clearance they have, only after deciding on some rational basis that the recipient has a legitimate need to access the information. The difficulty is in deciding what needs are legitimate and which aren't. The three reasons given above seem to be why this is done.) Compartmentation, in addition to perhaps making some of the above easier to accomplish, appears to be employed for the following further reasons: (note that deciding to create a new compartment or even declaring that a given piece of information falls within an existing compartment are not actions to be taken lightly: both entail considerable work and impose a burden on potential users of the information. That work and burden are presumably only incurred because the act of compartmenting a piece of information has some significant benefit above and beyond the mere fact of classifying it.)

Damage Assessment: to make it easier to make a damage assessment. Assuming all the controls are sufficiently effective, once someone has turned traitor it is in principle easier to limit the scope of the investigation to only those compartments that he had access to. This is in far contrast to having to determine exactly which of all, say, SECRET information one *actually* accessed — the effectiveness of an investigation of the latter assumes the manual and electronic audit trails work and give meaningful information, which is probably not a realistic assumption. (Most TCSEC audit trails are useless: suppose I browsed a SECRET classified forum on a Multics system with "pr -text /SDI/"[6] — the audit trail would, I think, show that I looked at the whole forum, and not just those entries dealing with SDI.)

Awareness: to lesson inadvertent, inappropriate, or unwise disclosure by making the individual more aware of the consequences of that disclosure through the indoctrination briefing[7]. It is not clear, however, that this offsets the risk posed by exposing information in the briefing that might not otherwise need to be known, including that which identifies why the information is important (and thus tempting.) That balance has to be tough for the policy-setters to make.

True Segregation: to make it easier to implement what has been called "anti-aggregation." I am told there are cases where information of type A or of type B by itself is of "ordinary" sensitivity (e.g, SCI) but that the combination of the two is so much more sensitive that the number and kind of people who have access to both must be specially controlled, even though the number and kinds who have access to either one by itself isn't particularly sensitive (in the normal course of events). I understand there are cases where the aggregate is not labelled AB but in fact is placed in an entirely new compartment, say, C (that might even be the normal way of dealing with the situation). Formal compartmentation makes it possible to bookkeep what combinations of accesses like this any given person has and thus prevent this particular kind of aggregation problem.

Neutrality: to lessen the risk in granting access to classified information by taking into consideration more factors than a generic clearance or background investigation would, in particular, factors specific to the particular kind of information involved. ("Do or don't give Jews access to information about/from Israel;" "Only give scientific information to someone with an advanced degree in a relevant

---

[6] "print all entries whose text contains the string SDI."

[7] The briefing may include much more than that strictly necessary for security awareness; it is only the security awareness part we are talking about here. It may even be that the security implications are self-evident, especially in the context of a sub-compartment of some umbrella compartment, so that there really is no security awareness briefing *per se*.

subject.'') This is subject to the ''jury of one's peers'' phenomenon: is it better to have expert or interested parties involved or not? Would you rather have a Jew (of what kind?) on the Israeli or Arabian desk or not (assuming he were convinced our interests and Israel's were not in conflict?) In any case, compartmentation gives the knowledgeable, concerned, and responsible parties (e.g., an Office of Primary Interest) the option of establishing additional conditions, either ahead of time or *ad hoc*, for the granting of access, such conditions reflecting the nature of the information involved as well as any other relevant information. Ordinary ''need-to-know'' cannot do so since there is no readily enforceable mechanism for ensuring that the person responsible for granting or denying access knows what considerations other than those included with or implied by the information itself ought to be examined. Note that the purpose here is *not* to institute extensive further background investigation procedures to ensure that the candidate for access approval is not in fact an agent of a hostile intelligence service, but merely to make prudent judgments to avoid humanly irresistible temptations and biases that might surface in themselves or be exploitable by a hostile intelligence service.

Judgement: to make it clearer to the people involved what improper access would be, i.e., who really has or should have ''need-to-know''. Anyone handling compartmented information knows that it is at least improper to give such information to someone who does not have formal access approval for it; someone handling ordinary non-compartmented information has only his own experience, training, and instructions to rely upon — there is no independent, external, authoritative source he can rely upon to help him decide who, once they have the necessary national clearance, should have ''need-to-know.'' Given proper initial indoctrination, crossing compartment boundaries could show up on a polygraph (and this potential consequence would be made known as part of the indoctrination) but no such impediment could be made to stick with ordinary need-to-know, even with the polygraph.

Human Error: to lessen the chances that someone will inadvertently not follow proper security discipline; this is not strictly a property of compartmentation *per se*, although the presence of formal labels helps, but is, in this discussion, more related to what a computer system does to help the honest person be honest.

Management: it appears that in some (perhaps many) cases the concept of compartment, as used here, is nearly equivalent to that of a ''project,'' where the latter is used here (informally) to refer simply to some kind of formally identified and managed activity, either a sensitive design effort, collection effort, analysis effort, or covert operation. (''Program,'' ''study,'' ''operation,'' etc. are other similar terms.) In such cases the concept of compartmentation is sometimes used in a variety of ways as a means of management control only indirectly related to the problem of controlling information about or from the project. Funding actions, management records, chains of authority (who can release information; who can authorize particular activities) may all be tied to the ''project'' in a manner that has security implications (especially when ''security'' is used in its broad meaning as encompassing confidentiality, integrity, and availability).

Threat Model and Vulnerability Assumptions

In a compartmented mode system there are two potential threat sources: all authorized users of the system and external agents of a hostile intelligence service (HIS). The authorized users would in most agencies and departments of the intelligence community in fact uniformly have a clearance that is even higher than the minimum required by DCID 1/14.

An authorized user can become an actual threat either by accidentally releasing information or performing some insecure action or by deliberately having become the witting or unwitting agent of an HIS. Although the likelihood of the deliberate threat by an authorized user is rendered extremely low by the personnel security practices of the intelligence community, it is non-zero, and increases as the number of users on a system increases. An authorized user who becomes a threat can either compromise security directly by releasing information he has access to or indirectly by finding and exploiting a security vulnerability in the system. In a C2 or B1 system it is essentially certain that there is at least one easily-found and exploited vulnerability[8] that would permit a malicious authorized user to have undetected access to *all* information in the system[9], regardless of what compartments it is in and what access approvals he has; it is for this reason that a C2 or B1 system does not help achieve many of the goals of compartmentation since any boundaries between compartments supposedly put in place by such a system are easily circumvented.

An external agent of a HIS can become a direct technical threat to a system by introducing malicious software or even hardware into the system. Such hostile system components could be inserted at several points of an information system. These components could then adversely affect the correct usage of the system without the knowledge of the authorized user. On any large system it must be assumed that a well-intentioned HIS would be successful in introducing such a malicious capability, although it might take time and patience. This is because large portions of a system, whether they be the operating system itself, commercially-available application packages (such as a data management system), or convenient utilities are written, maintained, and delivered by uncleared people through insecure channels. Any such malicious software operates with all the privileges of its user(s) and hence on a C2 or B1 system would also be able to access any information in the system (exploiting the vulnerabilities mentioned above.) The only additional facility such an attack would require is some means of getting the information the malicious software has accessed out of the system (since we are assuming in discussing external threats that all users continue to be trustworthy.) Since most intelligence community systems have *some* connection to the outside world (direct or via other systems), if only through nominally secure "message" systems handling unclassified traffic, or produce *some* nominally unclassified output, there is almost certain to be some means by which a malicious program can signal classified information out of the system hidden in what appears to be unclassified output or communications.

Although a B2 system is not assured to be completely without an exploitable flaw, including exploitable covert channels, the likelihood of them, and of being able to successfully introduce a malicious program that exploits them, is much less than in a C2 or B1 system. It is also the case that in order to compromise *all* of the information in a B2 system the HIS would either have to introduce malicious code into the operating system itself or into enough programs, or popular enough programs, that the code is run by at least one user cleared for *each* compartment; this is likely much harder to accomplish than the task on a C2 or B1 system where all that has to be done is to get the malicious code somewhere that at least *one* user, any user, executes it. A corollary of this is that if the HIS is targeting a specific compartment (e.g., project) on a given B2

---

[8] All such systems are vulnerable to Trojan Horses: programs, or possibly even malicious hardware, run by authorized users that unknown to their user attempt to bypass security controls, either by giving improper access (on a C2 system) or by exploiting covert channels (on a B1 system.) In addition, since C2 and B1 systems have little rigorous attention paid to design and implementation correctness, they are almost certain to simply have protection flaws that can be directly or indirectly used to permit unauthorized access to information by a technically knowledgeable person (or one acting under the direction of one, or running a malicious program written by one.)

[9] See the Appendix for a justification of this characterization of the B1 level of trust.

system it must insert malicious software in such a place that someone having access to that compartment will execute it — not any user will do, in contrast to a C2 or B1 system.

C2 vs. B1 vs. B2

Given the above ten goals of compartmentation, Table I below indicates which of the evaluation classes C2 through B2 help significantly to accomplish that goal in the face of the postulated threats. In the table "yes" indicates that the given level of system significantly helps accomplish the goal, "no", that it does not. N/A ("not applicable") means that that particular purpose of compartmentation is not something that a computer protection system can help much with. Notes on some of the table entries follow it.

TABLE I

Ability of "Trusted Systems" to Accomplish Compartmentation Goals

	C2	B1	B2
Damage Limitation	no	no	yes
Temptation	yes	yes	yes
Attractiveness (1)	no	no	yes
Damage Assessment	no	no	yes
Awareness (2)	N/A	N/A	N/A
True Segregation (3)	no	no	yes
Neutrality (4)	N/A	N/A	N/A
Judgement	N/A	N/A	N/A
Human Error (5)	no	yes	yes
Management (6)	no	no	yes

(1) A technically sophisticated hostile intelligence service would realize that anyone, or any program they ran, on either a C2 or B1 system could have access to all information on the system, perhaps given some technical help.

(2) The presence of system-enforced labels (occurring only on B1 or higher systems) does not seem to make a lot of difference here, although its perceived intrusion into previously routine operations would constantly remind someone of his responsibilities.

(3) The assumption here is that one is not especially concerned about temptation but whether a person, or a program he has run, has in fact been improperly granted access to the aggregate; a B2 system would be much better than a B1 system in ensuring that someone granted access to part of an aggregate (compartment "A"), or any programs he ran, did not have access to the rest (compartment "B").

(4) On the assumption that we are not attempting to weed out *a priori* malicious people, but rather simply to eliminate those who might be subject to unbearable pressures by virtue of the kind of information involved.

(5) A common occurrence, unfortunately, in a computer system, especially one tied to an electronic mail system, is accidentally sending information or a message to the wrong person. A B1 or B2 system at least prevents information from accidentally going to someone without formal access approval for it; a C2 system does not.

(6) Unless the B2 system implemented some form of rigorous (mandatory) integrity controls, the primary reason a B2 system would help with this goal and a B1 or C2 would not is that the core of a B2 system must be built to be more robust than that of a B1 or C2 and hence would be less vulnerable to malicious attempts inspired by a HIS to subvert management wishes. In short, if one believes routine data processing helps with project management and control, no additional benefit is gained by having a C2 or B1 system, rather than an unrated system, but having a B2 system does bring additional help in defending against deliberate attempts to thwart that management (either by authorized but malicious users or by software introduced from outside). A B2 system with rigorous integrity controls would in addition directly provide strong assurance that data (information in a data base, historical records, management directives and commands) could not be created or altered by unauthorized people or programs. A particularly insidious threat would be a malicious program that slowly and gradually "eroded" information over time, at a slow enough "rate" that it would not be routinely noticed; a B2 system would impose serious barriers to the extent of damage such a malicious program could cause by limiting its scope of activity to only those projects its users were authorized to change information in.

Conclusions

Where angels fear .... Of all the reasons for compartmentation for which the level of evaluation seems to matter, a C2 system helps with only one, a B1 with two, and a B2 with seven. If one were to live with only a B1 system one would have the following vulnerabilities to unauthorized disclosure not found if one were on a B2 or higher system:

- all users could be assumed by a hostile intelligence service to have reasonably easy access to all information on the system, perhaps given some guidance, regardless of what security level or compartment the information had, and thus all would be equal (and attractive) targets, either directly or by virtue of the programs they used.

- once a user were turned, either as a walk-in, target, or mole, he would have reasonably easy access to all information on the system, regardless of what security level or compartment the information had and thus any turned user could be assumed to be able to extract anything of interest. Any malicious programs introduced into the system must be assumed to be able to access all information in the system.

- sets of information which were supposed to be kept separated because their aggregation, conjunction, or juxtaposition would lead to grave consequences could not be assumed to be so separated once someone having access to any one of the sets turned sour, or once any such person had executed a malicious program.

- once a user is suspected or known to have turned bad, especially under the tutelage of a technically competent hostile intelligence service, it cannot be assumed that the only information he had access to and compromised is that which he had been given tickets for. Similarly, if a malicious program *were* discovered, even if it were known (or reasonably assumed) which users were likely to have run the program one could not assume that the only information it could have compromised was that for which the user had legitimate access.

Note that in most of the above the term "user" or "person" can almost always be replaced by the term "system" in a networking context. It must generally be assumed that in a C2 or B1 system anything any one of its users (or programs they run) can do, can be done by any other user (or program he runs), and this includes any actions that might be taken on behalf of that user (or programs he runs) on some other system it is connected to. This means that the

potential threats to a system include not only all its authorized users, or means by which malicious software might be introduced to it that they could be induced to run, but also *all* users (and their programs) in any C2 or B1 systems to which it is connected[10].

It's not for us mere mortals not privy to various higher matters of state to make policy, but it sure seems based on the above that running a compartmented mode system on only a B1 base is on pretty shaky ground: the only thing you gain over a C2 system, which in itself only serves to limit the scope of innocent or ambitious browsing, is preventing human errors. If there is any reason to believe that one is the target of a hostile intelligence service, or that any of one's technically competent users have any reason to believe they have something financial, moral, or political to gain by dealing with such a service, or other entity operating at cross-purposes to the interest of the U.S. Intelligence Community, one has to insist on B2 as soon as possible.

Acknowledgements

This note has been in various stages of preparation since late 1989. Before being submitted to this conference it was informally reviewed by a number of colleagues, all of whom shall remain anonymous since some requested to be. They, as well as the conference referees, made many useful suggestions, including the final version of the title, that have been incorporated and for which I am grateful. One of the conference referees urged that the scope of the paper be broadened to include a similar analysis of why B1 should not even be used in limited multi-level mode either, even though that is permitted for a risk range of CONFIDENTIAL to SECRET. I agree with the sentiment but have to reject the suggestion, both because of the lack of time and because on first glance I don't believe the argument, one way or the other, can be made as clearly as it can here.

References

[1] DCI Security Committee, "Security Manual for Uniform Protection of Intelligence Processed in Automated Information Systems and Networks" (U), supplement to Director of Central Intelligence Directive 1/16, 19 July 1988. (SECRET REL SEL FORN GOV) — all portions quoted, referenced, alluded to, or paraphrased in this paper are of course to sections, paragraphs, or tables explicitly marked as UNCLASSIFIED.

[2] DCI Security Committee, "Minimum Personnel Security Standards and Procedures Governing Eligibility for Access to Sensitive Compartmented Information," Director of Central Intelligence Directive 1/14, 13 May 1976 (UNCLASSIFIED)

[3] DoD Computer Security Center, "Computer Security Requirements — Guidelines for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments," CSC-STD-003-85, 25 June 1985.

[4] James P. Anderson, "An Approach to Identification of Minimum TCB Requirements for Various Threat/Risk Environments," *Proceedings of the 1983 Symposium on Security and Privacy*, Oakland, Cal., IEEE Computer Society, April 25-27, 1983, pp. 102-104.

[5] Department of Defense, "Security Requirements for Automated Information Systems", DoD Directive 5200.28, March 21, 1988.

---

[10] This note is already getting too long to amplify on this topic; suffice it to say that networking, even to a "small" extent, greatly expands the threat sources and means of exploitation of vulnerabilities.

[6] Department of Defense, "Automated Information Systems," *Industrial Security Manual for Safeguarding Classified Information*, Chapter 8, DoD 5220.22-M, January 1991.

[7] G.H. Nibaldi, "Proposed Technical Evaluation Criteria for Trusted Computer Systems," M79-225, Mitre Corporation, Bedford, Mass., 25 October 1979.

[8] A.-M. Discepolo, "Proposed Technical Evaluation Criteria for Trusted Computer Systems", Mitre Corporation, Bedford, Mass., MTR-8481, 30 September 1981.

[9] DoD Computer Security Center, "Trusted Computer System Evaluation Criteria," (DRAFT), 24 May 1982.

[10] DoD Computer Security Center, "Trusted Computer System Evaluation Criteria," (DRAFT), 15 November 1982.

[11] R. R. Schell, "Evaluating Security Properties of Computer Systems," *Proceedings of the 1983 Symposium on Security and Privacy*, IEEE, April 25-27, Oakland, Cal., pp. 89 - 95.

[12] D.J. Edwards, "Trusted Computer System Criteria: Classes B1 Through B3," *Proceedings of the Sixth Seminar on the DoD Computer Security Initiative*, National Bureau of Standards, Gaithersburg, Md., November 15-17, 1983, pp. 24-26.

Appendix

In Defense of the B1 Characterization Given Here

It is with some trepidation that I have included in this note a specific subjective characterization of "how good" a C2 or B1 system is since it can, not unreasonably, be taken as denigrating the hard work of all those who have produced B1 systems, including the several vendors involved in the DIA CMW project. Although I believe very strongly that the notion that a B1 system is not much more than "C2 with training wheels" has been generally accepted within the computer security community for a long time, try as I might I have (perhaps not surprisingly) been unable to find any professional or official writing that has stepped out and said so. Neither Schell [11] nor Edwards [12], the earliest formally published descriptions of the evaluation classes and levels, subjectively characterizes them in terms of ease of penetration, although Edwards' description of B1 contains a hint of the difference between C2 and B1 in a summary: "The big change at level B1 is the introduction of a mandatory security policy and supporting information sensitivity labels .... we are seeking a computing cultural revolution by making sensitivity labels an important, user visible part of the computer system." [12, p. 25] The closest to an authoritative reference I have been able to find is in the 1979, 1981, and 1982 precursors to the TCSEC:

> "Although extensively tested, a level 2 system [the precursor of B1] is still subject to design and coding errors. Testing should detect any obvious flaws; yet subtle ones might linger, to the advantage of untrusted users who are in a position to exploit them." [7, p. 25]
>
> "A level 2 system, like a Level 1 system, is intended for a generally benign environment... [it] would be suitable for DoD system high mode of operation." [8, p. 32]
>
> "A class <B1> system provides nominal mandatory ... access limitations ...." [9]

"A class (B1) system provides ostensible mandatory ... access limitations ...." [10]

In trying to track down the origin of the "training wheels" characterization I have received anecdotal evidence that Dan Edwards first used the term in that context in the summer of 1983, but no-one has been able to find it in writing. There is considerable additional private and semi-private evidence, some in the form of contemporaneous messages and letters, some as later recollections, attesting to the fact that the *-property was included in B1 (at the same time as what is known as the "simple security property") primarily to get applications programmers and system developers used to the constraints that would be imposed on them were they to move to a system that had credible assurances, but that too never showed up in any formal publication.

Finally, a fairly long attempt to characterize B1 in terms of what could and could not reasonably be expected of it was entered in the "Criteria" forum on the NCSC DOCKMASTER system in April 1987. Following is an abridgement of that entry:

> "...a B1 system is not expected to be able to enforce access controls in the face of any conscious competent technical effort to defeat them. ... A B1 system also cannot be expected to defend against all Trojan Horse attacks since it almost certainly will have covert storage channels and may even have direct channels through system objects not controlled by the TCB. ... It has not been subject to protracted penetration testing and based on past experience would succumb to a moderate (less than six man-months) attack by someone who has the opportunity and background to study the source code of the system. Although all known design or implementation flaws have been corrected, there has been no systematic effort to search for others (e.g., time-of-check-to-time-of-use on system calls.) There is no effective assurance that bugs, patches, or trapdoors have not been or cannot be implanted in the system during its manufacture or distribution."

Some 175 people eventually are recorded as having read that entry. A follow-up entry a few weeks later, which also was recorded as having been read by roughly the same number of people, drew the readers' attention to the fact that there had been no response to the characterization. As of this writing, there still has been no objection, which is part of the reason that I believe the characterization is accepted as fairly expressing the opinion of the computer security community[11].

---

[11] After this paper had been submitted for the conference an entry was posted in the NCSC DOCKMASTER "Trusted Product Evaluation Program Process Improvement" forum specifically addressing the issue of how good C2 and B1 systems are. (TPEP_Process_Improvement forum entry [0023], 5/18/92.) I do not know if the author wants his remarks attributed publicly, and forgot to ask before the revision deadline, so here they are without attribution:

> "As to C2 and B1, they are far too costly to develop/evaluate. We add process and meaningless steps such as design documentation to systems that are fundamentally bug-prone monoliths. Because that's what the systems are, the added effort results in no added security; C2 systems and B1's are about as secure as anticipated 8-9 years ago, but vastly more costly to develop.... B1 is no worse than C2 but no better. B1, by the way, was intended as "training wheels" for application developers who would develop to live with the *-property then transparently port their applications to B2 and above."

98 people, including several builders of B1 and C2 products, have read the entry and there have been no objections.

OPERATING SYSTEM SUPPORT FOR TRUSTED APPLICATIONS

Richard Graubart*

The MITRE Corporation
Burlington Road
Bedford, MA
01730

ABSTRACT

Trusted operating systems are finally becoming commercially available. But it is the user applications executing on trusted operating systems that actually address users' operational needs. To provide effective security, these applications often must be able to draw on the security features of the trusted operating systems, and the trusted operating systems must be able to provide certain security capabilities to the applications. In general, very little work has been done in the area of determining what security capabilities are required of a trusted operating system to support trusted applications. This paper provides some suggestions regarding how trusted operating systems could better support trusted applications.

INTRODUCTION

Trusted systems, in particular those that can operate in multilevel or compartmented mode, have recently started to become commercially available. Trusted applications, such as trusted mailers, trusted editors, and trusted Database Management Systems (DBMSs), are also being developed. Trusted application technology lags somewhat behind trusted operating system technology, if only because trusted operating systems are required to serve as a base for the trusted applications.

The design and implementation of trusted applications does not merely entail selecting arbitrary applications, placing them on a trusted operating system, and watching them operate correctly. The applications must be carefully integrated into the trusted environment provided by the operating system. For its part, the operating system must provide certain security capabilities to support the trusted applications. In general, very little work has been done in the area of determining what security capabilities are required of trusted operating systems to support trusted applications. Documents such as the Trusted Computer Security Evaluation Criteria (TCSEC) [DOD85] do mandate certain minimum requirements on trusted operating systems. However, these are requirements for ensuring that the trusted operating system is secure. Trusted operating systems can be built that are in compliance with the TCSEC, but still do not necessarily provide the necessary security capabilities needed by a trusted application.

The purpose of this paper is to provide some insight and suggestions as to what security capabilities could be incorporated into trusted operating systems that would make them a better base for trusted applications.

---

* This paper represents the views of the author, not necessarily those of the MITRE Corporation.

TRUSTED APPLICATIONS: WHAT ARE THEY AND WHY ARE THEY?

Before discussing the operating system needs of trusted applications, we need to discuss exactly what we mean when we refer to trusted applications. By an application we mean a set of software that performs some specific set of functions (e.g., a DBMS, a mailer, etc.) for the purposes of addressing some specific problem (the need to interrelate data, the need to pass information between users, etc.). A trusted application is one that performs some specific security task while satisfying a needed operational function. We do not consider an application trusted if its security functionality is performed entirely by the underlying operating system. For our purposes an application is considered trusted only if the application itself performs some security related function. The nature of the security functionality performed by the trusted application can vary; it can be access control, it can be labeling information, it can provide some supplemental authentication capability, etc.

Trusted applications are required when the operational needs of the application cannot be accomplished (at all) or cannot be accomplished securely even with the support of the trusted operating system. As an example, a DBMS may require labeling and mandatory access control (MAC) at a level of granularity not provided by the underlying operating system. To satisfy this need, a trusted DBMS may be required which will enforce labeling and MAC at a granularity finer than that of the operating system.

For trusted applications to perform their tasks it may be necessary for the application to override some policy of the underlying operating system. As an example, a trusted DBMS may need to support its own MAC and labeling policies on the objects under its control. The labeling or access control decisions performed by the trusted DBMS would be independent of the access mediation provided by the trusted operating system. In order for the application to perform, such actions require some privilege or set of privileges from the underlying operating system that allow it to operate independently of some of the policies enforced by the operating system. Continuing with the example, a trusted DBMS may require a *Violate-MAC-Policy* privilege, or a *Relabel-Data* privilege in order to label data records and perform access control on the records.

Not all trusted applications require privileges, only those whose task requires them to override the policy of the underlying operating system. Some applications may enforce a policy that is orthogonal to that of the underlying operating system. For example, a trusted DBMS could exist that does not enforce any MAC or labeling, but does enforce some entity integrity policy.[1] Such a policy is independent of the policies enforced by the underlying operating system.

Thus, trusted applications are required when a particular problem needs to be addressed in a secure manner, and for some reason the application cannot rely on the underlying operating system to provide the necessary security support. As noted above, there are different types of trusted applications. Some trusted applications need to override the policy of the underlying operating system, others enforce policies that are completely independent of those of the operating system. Some trusted applications both override some policy of the underlying operating system, and also enforce some policy that is orthogonal to those policies enforced by the underlying operating system. In the next section we discuss features that, if incorporated into the underlying operating system, would make the implementation of trusted applications easier. Most of the proposals are directed at supporting applications which override the policy of the underlying operating system.

---

[1] Such a policy would allow the DBMS to enforce constraints on what data values could be inserted in a given field. As an example, it would ensure that in a salary field no negative salaries may be inserted.

This is because we believe these types of trusted applications to be the most common type of trusted applications, and the ones for which operating system support is the most needed. However, some of the suggestions can lend themselves to both types of trusted applications.

NEEDED OPERATING SYSTEM SUPPORT FOR TRUSTED APPLICATIONS

There are a variety of ways for a trusted operating system to provide support for trusted applications. One possible way is for the operating system to provide some security capability that the trusted application depends upon to successfully carry out its mission.

Another way is for the operating system to provide support that will help limit the possible complexity of the trusted application. If the trusted operating system provides a needed trusted capability, and does so in a manner that is highly flexible, then this may eliminate the need for the trusted application to provide the trusted capability on its own. This would limit the complexity of the trusted application, and in so doing likely lessen the cost of the application and speed up its development time.

Finally, it must be noted that those trusted applications which implement policies that are not independent of the underlying operating system have the potential of interfering with the operations of the underlying operating system. Not only must such trusted applications be scrutinized, but because of the possibility that the applications may interfere with the underlying trusted operating system, the operating system itself may need to be reevaluated. Such reevaluation is time consuming. In addition, it undercuts the utility of employing evaluated trusted operating systems, as the already evaluated operating system may require reevaluation even when no changes have been made to the operating system itself. If the operating system could be designed in some way that would lessen or eliminate the need to reevaluate it when used in conjunction with a trusted application, this would further support trusted applications.

In the remainder of this section we describe some possible enhancements to trusted operating systems, all of which we believe in one way or another provide support for the use of trusted applications.

SUPPORT FOR TRUSTED PATH

Many trusted applications need to be able to communicate with a user in a manner that clearly and unambiguously indicates that the user is interacting with the trusted application and not some untrusted software. The traditional means of establishing such communications is via a trusted path. It is not desirable for the trusted application to provide the trusted path on its own. Trusted paths often require some control of the hardware, and only the underlying trusted operating system should have access to the hardware. Therefore, what is needed is for the underlying operating system to make its trusted path mechanism[2] available to the trusted application. This would allow the trusted application to connect its trusted path mechanism with that of the underlying operating system, thus ensuring an unbroken, unspoofable trusted path from the user to the TCB.

This trusted path should be bidirectional, that is, either the user or the trusted application should be able to invoke it. A primary reason why the trusted path needs to be bidirectional is to accommodate the highly interactive, real-time operations of many trusted applications. As an example, it is possible to have a DBMS environment where a user issues a query, then performs some other action while waiting for the TDBMS to respond to the query. When the TDBMS does respond, it is important for the user to know that the response has come from the TDBMS, not

---

[2] The TCSEC requires a trusted path for systems at B2 or higher.

some untrusted code that is attempting to spoof the user. Having the TDBMS be able to invoke the trusted path provides a means of ensuring the user that the communication did indeed come from the trusted DBMS.

The operating system trusted path could be made available to the trusted application via system calls or library routines. For additional control, only appropriately privileged trusted applications would be allowed to utilize these system calls or library routines. Note that none of these proposals (use of library routine or systems calls, use of privileges, bidirectional trusted paths) is beyond the current state-of-the-art. Nor do any of these proposals conflict with the requirements of the various security metrics (e.g., the TCSEC). However, because these capabilities are generally not called for in the various security metrics,[3] most trusted operating systems do not include these capabilities.

IMPROVED PRIVILEGE SUPPORT

As noted earlier, in order to perform their functions, many trusted applications require some privilege from the underlying operating system. We propose that the privileges provided by the operating system should be at as fine a granularity as possible. The use of fine granularity of privilege is consistent with the TCSEC concept of least privilege.[4] In addition, we believe that the use of fine-grained privilege may help minimize the reevaluation of the underlying operating system. For example, if a system only supports a single 'super-user' privilege that overrides all of the underlying operating system policies, then an application that only requires the ability to write to the operating system audit trail will be given the ability to override the underlying MAC, and Discretionary Access Control (DAC) policies, as well as the ability to override the system audit policy. Such action clearly violates the concept of least privilege. In addition, the entire operating system must be reexamined to ensure that it works properly in conjunction with the trusted application. But, if the underlying operating system supports a *Generate Audit Data Privilege*, then only that part of the operating system responsible for enforcement of audit policy would need to be reexamined.

It is very difficult to define what is the appropriate minimum set of privileges that an operating system should enforce without adversely impacting the ingenuity and flexibility of the operating system vendor. However, wherever possible, the operating system should enforce as fine a granularity of privilege as possible (many trusted operating systems already provide a very fine granularity of privilege). The set of privileges provided by a trusted operating system should take into account likely requirements of trusted applications. The choice of privileges should also take into account the requirements against which the operating system will be evaluated. At a minimum, privileges should not cross policy boundaries. In addition, within a given policy, vendors should attempt to ensure that capabilities that map to different requirements should not couple together in a single privilege [CMWEC]. We believe that such actions, if implemented, would not only ensure that privileges conform to the concept of least privilege, but in so doing would help minimize the amount of reevaluation required of a trusted operating system.

---

[3] The Compartmented Mode Workstation Evaluation Criteria, DDS-2600-6243-91 [CMWEC], does require the operating system provide a bidirectional trusted path that can be made available to trusted applications.

[4] At the B2 level the TCSEC requires the use of least privilege but does not elaborate on the concept.

In addition to supporting fine-grained privileges, operating system support for privilege bracketing would also aid trusted applications. Without privilege bracketing, an entire process is granted the capabilities associated with a privilege. The use of privilege bracketing constrains where in a process a privilege may be employed. As a result, only the code that exists between where a privilege is activated and deactivated can employ the privilege. In no other locations within the process can the privilege be invoked. Having the operating system provide this capability further encapsulates the actions of a trusted application, presumably limiting the amount of the trusted application[5] that needs to be scrutinized for correct operations and use of privilege.

ENCAPSULATION OF TRUSTED APPLICATIONS

One of the issues with incorporating a trusted application with an already evaluated trusted operating system is that the trusted application has the potential to interfere with the operations of the trusted operating system by applying the privileges it was granted to resources under the control of the operating system. For this reason, the addition of a trusted application to an already evaluated operating system requires that some portion of the underlying operating system must be reevaluated. The use of fine-grained privileges provides some limits on the amount of the operating system that needs to be reevaluated. To further limit, and possibly eliminate, such reevaluation requires some means of encapsulating the actions of the trusted application so that it cannot interfere with the trusted operating system.

One means of encapsulation is for the trusted operating system to enforce some typing policy and associated mechanism. Under such a system the operating system would ensure that all applications trusted and untrusted (indeed all subjects and objects) would have some type associated with them. Thus, a DBMS (and associated subjects and objects) might have a DBMS type associated with it, a Mailer would have a Mail type associated with it, etc. The object typing policy would ensure that an application of one type could only access subjects and objects of some specified type. Some systems already support such capabilities (LOCK [SAYD87], XTS-200 [HFSI92]). The most restrictive variation of this policy would be to ensure that an application could only access subjects and objects of the same type. By imposing such restrictions, one could ensure that a trusted application could not interfere with the actions of the trusted operating system or some other trusted application.

An encapsulation mechanism as proposed above would constrain which subjects and objects a trusted application could access, and, hence, would ensure that trusted applications could not interfere with the workings of trusted operating system's trusted applications. This satisfies the security concerns but can cause some operational difficulties. Operationally, there will usually be some entities of one type that need to be accessed by entities of another type. For example, it is likely that most trusted applications will require access to the operating system clock. But, if the encapsulation policy is such that an application can only access entities of the same type, then trusted applications will be prevented from accessing the clock which would be of a different type. An alternative is to enforce a less rigid policy which would allow certain objects to be accessed by subjects of specified types or permit specified subjects to access objects of multiple types. The difficulty with these alternatives is that they weaken the encapsulation policy, and, therefore, one can no longer be ensured that the actions of a trusted application are constrained to only entities of its own type. Thus, these more permissive encapsulation policies while allowing greater operational flexibility, and greatly reducing how much of an application would need to be re-evaluated, would not completely eliminate the need for reevaluation of trusted applications.

---

5 Privilege bracketing is only effective if certain other constraints are taken into account, such as prohibition of GOTOs, or self-modifying code in the application.

LABELING AND MAC CONSISTENCY

One of the primary reasons for the development of trusted applications is the need to enforce fine grained labeling and MAC at a level of granularity finer than that enforced by the operating system. Trusted DBMSs often need to label at the tuple or record level, trusted editors may need to label at the paragraph level, etc. The granularity provided by the operating systems (e.g., file level) is generally not sufficiently fine to satisfy the needs of these trusted applications. Some architectures have attempted to address this problem by aligning the objects of the trusted application with that of the operating system [DENN87]. Thus, in the case of a DBMS, all of the Secret tuples of a relation would be in a Secret file, all of the Top Secret tuples would be in a Top Secret file, etc. For many environments this is an acceptable solution. However, for systems which require an extremely fine level of granularity and a large number of different security levels, this architectural approach may not be optimum. This architectural approach requires the physical partitioning of the application objects into multiple operating system objects with the result that logically related application objects (e.g., data tuples of the same relation, but of different security levels) are no longer in physical proximity. As noted in [GRAU89, GARV89], this may result in performance dropping logarithmically as the number of security levels increases.

For the performance reasons cited above, many trusted applications often perform their own MAC and provide their own label manipulation and label conversion. For some systems these label and MAC checks are quite simple and the amount of code involved is relatively small. However, in other systems these label checks can be quite complicated. For applications running on these systems, duplication of such label and MAC checking code in the application could prove to be expensive. In such environments it is quite beneficial if the labeling and MAC checking code is made available to the trusted application. This could be accomplished by placing such code in library routines or providing appropriate system calls to such code. In either event, the code would only be accessible to the appropriately privileged trusted application. Making such code available to the trusted application reduces part of the burden on the trusted application, making the applications somewhat less complicated and, hence, less expensive to build and evaluate. In addition, such an approach also ensures that the applications and the operating system are using labels consistently, thus eliminating the need to convert labels when data is passed between the operating system and the trusted applications.

Still, another way that a trusted operating system can relieve some of the labeling and MAC burden from a trusted application is for the trusted operating system to provide an arbitrarily fine level of labeling granularity (e.g., down to byte).[6] By having an operating system provide such fine labeling granularity, then the trusted application no longer needs to provide its own labeling, but can instead rely on that of the operating system. Because the security granularity could be dynamically selected by the trusted application, application objects (e.g., database records, mail messages) would not need to be physically partitioned into separate operating system files as described earlier. This lack of physical partitioning would mean that system performance would not be impacted by the number of required security levels, as is the case when operating system granularity is more coarse. Because the trusted application would not need to enforce its own labeling or MAC, the cost and complexity of the application would likely be less, and the evaluation/certification of the application would be easier than if it had to enforce its own labeling an MAC.

---

[6] MITRE has enhanced a prototype CMW to provide just such a capability. See [PICC91] for further details.

CONCLUSION

As discussed above, one can categorize trusted applications into two broad categories: those whose actions are independent of the operating system and those whose actions are not independent of the operating system. We believe that the vast majority of trusted applications fall into the latter category. As we have also noted, there are three possible ways to categorize operating system support for a trusted application. There are those enhancements which allow the trusted application to perform some action it would not otherwise be possible to occur. There are those enhancements which limit the amount of reevaluation that is required of the operating system. Finally, there are those enhancements which limit the complexity of the trusted application. The enhancements described in this paper cover all of the categories.

The support for an extendible trusted path is necessary for any trusted application which requires an unspoofable connection between it and the user. Note that this enhancement is necessary regardless of whether the application is dependent or independent of the policies of the underlying trusted operating system.

As noted above, improved privilege support in the operating system can limit the needed reevaluation of the trusted operating system. In addition, the ability to use finer grained privileges, and privilege bracketing helps limit the size and complexity of the trusted application by providing for clearly designated areas of the application which need privileges and those that do not. Such enhancements are clearly intended for those applications which are not independent of the operating system, and, hence, can override the security policy of the operating system.

The use of object typing to encapsulate the actions of the trusted application is intended to limit reevaluation of a trusted operating system. The encapsulation enhancements are directed toward those applications whose actions are generally not independent of those of the trusted operating system.

The enhancements directed at ensuring labeling and MAC consistency are primarily targeted at simplifying the functionality of the trusted application by removing a burden from the application.[7] Such enhancements are clearly directed at applications that are not independent of the actions of the trusted operating system.

It is important to note that there are operating systems currently available that provide many of the already mentioned capabilities. Also, the enhancements discussed in this paper, trusted path, improved privilege support, the ability to encapsulate the action of trusted applications, providing a means to ensure labeling, and MAC consistency between applications and the operating system, are enhancements that we believe to be useful for supporting the applications we have studied. Other enhancements to the trusted operating system that could support trusted applications are likely. The nature of these enhancements will become clearer as we gain greater experience with trusted applications.

---

[7] The enhancements may also limit the need to reevaluate a trusted operating system since there is no longer any application MAC that can interfere with the operating system MAC.

LIST OF REFERENCES

[CMWEC] *Compartmented Mode Workstation Evaluation Criteria Version 1*, DDS-2600-6243-91, Defense Intelligence Agency, 1991.

[DENN87] Denning, D., E., et. al., *A Multilevel Relational Data Model,* 1987 IEEE Symposium on Security and Privacy, Oakland, CA, 1987.

[DOD85] Department of Defense *Trusted Computer System Evaluation Criteria,* December 1985, DOD 5200.28-STD.

[GARV89] Garvey, C., et. al., 1989, *A Layered TCB Implementation vs the Hinke-Schaefer Approach*, 1989 IFIP Workshop on Database Security, Monterey, CA.

[GRAU89] Graubart, R. D., 1989, *A Comparison of Three Secure DBMS Architectures,* 1989 IFIP Workshop on Database Security, Monterey, CA.

[HFSI92] HFSI, 1992, *XTS-200 Trusted Facility Manual.*

[PICC91] Picciotto, J., and D. F., Vukelich, 1991, *Fine Grained Labeling, Volume 1: Operating System Support*, MTP 387, Volume 1, The MITRE Corporation, Bedford, MA.

[SAYD87] Saydjari, O. S., et. al., September 1987, *Locking Computers Securely,* Proceedings, 10th National Computer Security Conference, Baltimore, MD.

Operational Support of Downgrading in a Multi-Level Secure System

Doug Nelson
Greg Factor
Jim Studt
Mary Yelton
Steve Heffern
Frank Kramer

MLS GDSS Program
Digital Equipment Corporation
721 Emerson Rd.
St. Louis, MO 63141

Keywords: Downgrade, Multi-level Security, Relational Database

Point of Contact: Doug Nelson (314)991-6232

Abstract

Downgrading is a complex and time-consuming process that is absolutely necessary to operate a Multi-Level Secure (MLS) system. This paper describes the requirements and design of a Downgrader Utility planned for use in the United States Transportation Command/ Air Mobility Command's (USTRANSCOM/AMC) MLS Global Decision Support System (MLS GDSS). The Downgrader Utility provides operational support to permit efficient and timely downgrading of classified information during execution of AMC missions.

Introduction

USTRANSCOM/AMC operates the GDSS command and control system to manage its airfleet. Currently, AMC operates an unclassified system and a separate classified system, both executing the same software. As part of its MLS testbed activity and with the support of NSA, DISA and several other Federal agencies, AMC is developing an operational B1-certifiable MLS system to replace the existing single level GDSS system [1]. The system is targeted for an environment that is hosted on a DEC VAX/Security Enhanced VMS (SE/VMS) system. A B1-targeted relational database managements system (RDBMS) is integrated into the system architecture to provide database management services.

As part of this development effort, requirements for downgrading operations were identified and a prototype design developed and implemented. The paper describes these requirements and some of the more important design aspects.

Key Requirements

Of the numerous functional requirements that must be satisfied by the prototype, several management and presentation requirements had a significant affect on the design and implementation of the downgrade utility. These four requirements are:

- Range of downgrading granularity
- Transaction integrity
- Support for polyinstantiation
- Operational simplicity
- Audit capabilities

Scheduling and execution of missions by the MLS GDSS must be performed in a manner that provides uncleared individuals with sufficient data to operate, while protecting the mission's classified data and preventing inferences about that data. The system accomplishes this, in part, by supporting cover stories. In order to limit the understanding of the mission and limit inference, fictitious or bogus information that is indistinguishable from real data is entered into the system at the unclassified level for uncleared users. Classified information representing the real plans of the mission are entered into the system for the secret users, along with some real unclassified data necessary to initially schedule the mission. The bogus unclassified data is called a cover story and hides the existence and values of secret data while limiting inferences about the classified aspects of the mission.

During the execution phase of the mission, secret data must be downgraded and made available to uncleared users in a timely fashion. This downgrading process does not include all classified aspects of the mission, but only those portions that are necessary for the uncleared individual to perform his/her duties. Through limited downgrading and the use of cover stories, the remaining secret data is protected from viewing and inference. For example, consider an aircraft fly-

ing a classified mission consisting of several stops at various airbases. The uncleared ground crews that service the plane after its landing must be made aware that the plane is arriving in advance in order to make preparations to service the plane. However, this information need not be made known to them until only a short period of time before the aircraft arrives. All other aspects of the planes mission, such its cargo or passengers, destination, and departure time remain classified and protected through the use of cover stories. Only when it is absolutely necessary is secret data downgraded to allow the execution of duties by the uncleared individuals.

In other cases, it is necessary to be able to downgrade nearly simultaneously a large number of classified details about several mission and aircraft operating those missions. This might occur during a large exercise or contingency when multiple missions operate together at the same time. Thus, the downgrading utility must be capable of downgrading a varying granularity of classified data, ranging from a single data element such as arrival time of a mission at a specified location through all classified schedule information associated with hundreds of aircraft executing a contingency plan. Since the downgrading of classified information is time critical during execution, the downgrading utility must be able to operate in a timely fashion to ensure that uncleared individuals are made aware of duties they must perform.

The MLS GDSS makes deliberate use of polyinstantiation to support cover stories to limit inference[2,3]. In a polyinstantiated database that uses cover stories, the actual process of downgrading information involves overwriting of cover story data at the unclassified level with data that was stored at the secret level prior to the downgrade. The downgrader utility must understand the use of polyinstantiation and cover stories to provide a mechanism to accomplish this action in a relational database.

The MLS GDS System also enforces, through its trusted computing base code extensions, aggregation rules. These rules control the classification of data elements in relation to the classification level of other associated data elements. The downgrade utility must also support and follow these same aggregation rules.

An important requirement for the downgrader utility is the preservation of data integrity within the database during the downgrade operation. Since the MLS GDSS system is used by hundreds of flight controllers and planners simultaneously around the world, it is important that the downgrading operation not disrupt the availability of the database, or corrupt the integrity of data associated with a particular mission. The actual downgrading of information stored in multiple table within the database should occur as a single complete transaction, making the new unclassified data set available in a single process. Operationally, it is very important that all of the chosen set of secret data to be downgraded become unclassified. Partial downgrading could inadvertently disrupt operations and open inference channels to uncleared individuals.

Design Overview

The solutions to the requirements identified in the prior section are reviewed and discussed in the following paragraphs. The downgrader utility uses trusted computing base extensions of the MLS GDSS to access and communicate with the database and to manage and protect sensitive data structure and associated labels [3]. In the system, cover stories and data element level labeling are provided through the deliberate use of polyinstantiation. Aggregation rules and clas-

sification determination are provided by the system and these rules are available to other trusted components, including the downgrader utility. Solutions to the downgrade requirements are implemented with trusted application software.

Range of Downgrading Granularity

The MLS GDSS database is a relational database that has been normalized to handle the over 1500 data elements that comprise the system. Entity relationship analysis defined the major objects stored within the database, along with the clusters of tables that hold the attributes associated with the objects. These objects consist of items such as MISSIONS, AIRCRAFT, AIRCREWS, and CARGO. Data structures within the downgrade utility store meta data about the tables that hold information about an object and how these tables are interrelated. This meta data is used by the downgrader utility to access all data attributes of an object for presentation to the user and during the downgrade operation. After viewing the contents of an object or multiple collections of related objects, such as all missions associated with an exercise, the user can select data elements for downgrading. This selection may entail all downgradeable elements associated with an object, such as a mission, or one single element, such as the landing time at a specific site for a particular mission. Using meta data, the downgrader utility accesses the database tables holding the chosen information and downgrades those data elements

Transaction and Access Integrity

One of the major drawbacks of existing downgrade capabilities supplied with COTS relational database products is the lack of transaction integrity and denial of access to the database objects being downgraded. Downgrading in some database products requires the creation by the user of additional temporary tables and the movement of classified data from the original tables to the the temporary tables. The temporary tables are then downgraded and the unclassified data moved back to the original table. During such a process, database integrity and accessibility may be compromised. The MLS GDSS downgrader utility performs downgrading operations in a manner that resembles other application transactions. The selecting and updating that takes place during the operation is done within the context of a single transaction and does not require the creation of intermediate tables.

Polyinstantiation Support

Downgrading support that exists in most COTS RDBMS products consists of rewriting of the security label of the tuple holding the classified data to be downgraded. In a polyinstantiated database that uses cover stories like MLS GDSS, this support is insufficient and provides little advantage. Downgrading in these products consists of updating the security label column with the value that the data is being downgraded to, e.g., UNCLASSIFIED. Unfortunately, this approach does not work in a database that makes deliberate use of polyinstantiation as the MLS GDSS does. Since the security label column is part of the unique key of the relations in the database[2], attempts to rewrite the security label of a polyinstantiated tuple could violate the uniqueness of tuples within the table. Instead of modifying the security label column, the downgrader utility performs the downgrade operation by updating data elements stored in the UNCLASSIFIED tuple with the values stored at the secret level in the SECRET tuple. The se-

cret values stored in the SECRET tuple are then removed (nulled) or the entire secret tuple is deleted. This operation requires the user of the utility to have the downgrade privilege.

Operational Simplicity

The primary reason for developing a downgrader utility is to provide the users with downgrade privileges with a tool that simplifies their duties and improves the efficiency of using and managing a system that supports multi-level data. Downgrading information stored in over 1500 data elements and over 200 tables is a complex task. Without the support of such a utility, the downgrader must be intimately familiar with the database schema, as well as the transactions that are performed by the application. In many cases, data fields stored within the database are not directly visible to the user through application screens. These fields may be used for internal controls and specialized functions performed by the application to interrelate data and maintain integrity. Without an understanding of the transactions, the downgrader might inadvertently overlook downgrading these fields, resulting in potential corruption of the database integrity. The downgrader utility through its meta data extensions is capable of finding and downgrading the appropriate data elements during an operation.

RDBMS Independence

The downgrader utility makes use of trusted computing base extensions present in the MLS GDSS. Because of this, the utility is relatively independent of the underlying RDBMS product. It was a design goal of the system to allow adaptation of the utility to future MLS applications that may form part of the MLS GDSS. Because the rules describing how to access all the components that make up an object are stored in meta data, extending the utility or modifying the utility is a simplified process. This design also reduces the amount of trusted code in the utility.

Auditing Extensions

The downgrader utility also provides auditing capabilities that extend beyond those provided by most RDBMS products that include downgrading features. One example is the lack of historical information captured during a downgrade operation. Most databases are capable of auditing the read and write operations that occurred during a downgrade and the tuples that were affected. However, once data is overwritten or deleted, it is impossible to reconstruct precisely what occurred. One of the requirements of the system that the downgrader utility provides is a historical view of the downgrade operations. Before and after snapshots of updated and deleted data are captured and recorded to an audit file. This permits a auditor to accurately reconstruct the data set before and after a downgrade, perhaps aiding analysis of inadvertent downgrades or deliberate attempts at disclosure.

Summary

This paper has described the requirements and design of a downgrading utility to support an operational MLS C2 system. The utility operates in conjunction with a polyinstantiated data-

base system that provides data element level labeling and cover stories. Key amongst the requirements was the ability to provide operational simplicity to aid in downgrading data ranging from a single data element to a large collection of related data stored in multiple database tables while maintaining data integrity and database access. The utility is designed to take advantage of meta data extensions that describe the major database objects and to be independent of any particular RDBMS product. The utility also is designed to extend the auditing capabilities of existing RDBMS products to aid analysis of downgrade activity. Finally, the paper points out some of the needs identified in an operational MLS C2 system to permit efficient and simplified downgrading to occur.

References

[1] S. Doncaster, M. Endsley, and G. Factor, "Rehosting Existing C2 Systems Into An MLS Environment", *Proceedings from the Sixth Annual Computer Security Applications Conference*, Tucson, Arizona, 1990.

[2] T. Lunt, "Polyinstantiation: An Inevitable Part of the Multilevel World", *Proceedings of the Fourth Workshop on Foundations of Computer Security*, Franconia, New Hampshire, 1991.

[3] D. Nelson and C. Paradise, "Using Polyinstantiation to Develop an MLS Application," *Proceedings from the Seventh Annual Computer Security Applications Conference*, San Antonio, Texas, 1991.

PM: a Unified Automated Deduction Tool for Verification[1]

George Fink (gfink@cs.ucdavis.edu)
Lie Yang (yangl@cs.ucdavis.edu)
Myla Archer (archer@cs.ucdavis.edu); (916)752-7583

University of California, Davis
February, 1992

Abstract. We are developing a tool called PM (for Proof Manager) that provides flexible support for the use of automated deduction in verification by providing a common interface to a variety of existing theorem provers. PM works to blend the strengths of these different provers into a proof system more powerful than its compenent parts individually. The assertion language of our PM prototype is that of the HOL theorem proving system [Gordon87], chosen because it provides a verifier with a very expressive higher-order assertion language that is especially suited to expressing specifications of complex systems (such as secure distributed systems) and high level reasoning, but—being essentially a proof checker—is tedious to use for lower level assertions that are simple for automatic first-order provers. PM is intended to supplement HOL in a sound manner with access to various existing, polished automatic provers. Simultaneously, PM provides a verifier with a convenient means of managing and summarizing proofs.

Keywords: verification, system verification, automated deduction systems

1. Introduction

For large verification projects, automated support is essential. In particular, automated support for reasoning about the many complex (if not usually deep) formulae generated as verification conditions in the process of formal verification is almost a necessity. Being involved in several ongoing hardware and software verification projects, we are interested in making this automated reasoning support as general and convenient to use as possible. This paper reports on the current status of and future plans for PM (Proof Manager), a tool we are developing to achieve this goal. The working prototype of PM is now being used in some of our current verification projects.

One feature provided by PM is an improved interface to an existing theorem proving tool that has proven to be appropriate for hardware verification, namely, the HOL (higher-order-logic) system developed by Michael Gordon and others at Cambridge University [Gordon87]. The HOL system has proved useful in both hardware verification [Cohn88, Joyce88, Win90] and other security applications such as the proof of the security kernel of an operating system [A-FL91].

The usefulness of HOL for large verification projects is the result of its provision for expressing and directly reasoning about higher-order assertions—that is, assertions that are general statements about functions. This capability makes the factoring of proofs natural in HOL, and thus can reduce the work involved both in individual top-down verification projects and in whole families of similar verification efforts. On a lower level, higher-order assertions are a natural way to encode properties of designs whose specifications involve the concept of time: e.g., hardware designs that express the relationship between values carried on various wires as a function of time. The strength of this temporal abstraction can be seen in the following specification of an ALU from [Joyce88]. (Note that in the definition, => and | correspond to if-then and else respectively, and the juxtaposition of expressions indicates function application, and finally that the result of function application—e.g., add rep—can itself be a function.)

```
Alu = |– ∀ rep f1 f2 inp1 inp2 out. Alu rep f inp1 inp2 out =
        ∀ t : time. out t = (((f0 t,f1 t) = (T, T)) => ((inc rep) (inp2 t)) |
```

[1]This research is supported in part by the Department of Defense under contract grant DOD-MDA 904-91-C-7052.

```
((f0 t,f1 t) = (T, F)) => ((add rep) (inp1 t,inp2 t)) |
((f0 t,f1 t) = (F, T)) => ((sub rep) (inp1 t,inp2 t)) |
((wordn rep) 0))
```

While the theorem-proving capabilities (in the sense of the appropriateness of the logic) of the HOL system are satisfactory, the standard interface puts a real burden on the system user, limiting access to subgoals and leaving details of the creation of a large proof from smaller steps up to the user. The HOL interface of PM relieves much of this burden by, for example, making it possible to keep track of steps of a proof by associating them with nodes in a proof tree, and in supporting various ways of combining short steps into a larger proof while compacting the tree, relieving HOL users of considerable work that would otherwise need to be done by hand.

There are trade-offs to consider in choosing the most appropriate form of support for automated reasoning in large verification efforts. As we have indicated, powerful theorem-proving systems such as HOL or Nuprl [Con86] that support higher-order reasoning lend themselves to a particularly straightforward formulation of many specifications, and reasoning about these formulations directly, as well as generalizing proofs. On the other hand, proofs in such systems as HOL and Nuprl must be guided in detail by the user; reusable tactics can be used to some extent, but even so, these tools more resemble proof-checkers than automated theorem provers. The tools become particularly frustrating to use when one reaches the level of simple lemmas whose proofs would be straightforward and largely automatic if one only had on hand, say, a simple resolution prover, an automatic induction capability, or a linear arithmetic decision procedure. To a varying degree, these capabilities can be approximated by special tactics (or by auto-tactics in Nuprl), but the approximations are generally either incomplete or less efficient than those provided by polished specialized systems.

Theorem provers with less powerful inference mechanisms, such as resolution provers or the Boyer-Moore prover [BM88] are closer to being automatic provers. Given an appropriately formulated valid first-order assertion as input, the Boyer-Moore prover will generally terminate with success or failure in finding a proof, and a full resolution prover will either terminate with success or will fail due to space considerations. In the following example, taken from [Joyce88], a lemma requiring a complex HOL tactic proves in a fraction of a second in the Boyer-Moore prover. In addition to the time taken in generating this tactic, HOL takes several seconds to produce a result.

Original:

$\forall$ m n p q. (m < p) $\wedge$ (n < q) $\Rightarrow$ ((m + n) < (p + q))

HOL Tactic:

```
let th1 = SPECL ["m";"p";"n"] LESS_MONO_ADD_EQ in
let th2 = SPECL ["n";"q";"p"] LESS_MONO_ADD_EQ in
let sublist = [SPECL ["n";"p"] ADD_SYM;SPECL ["q";"p"] ADD_SYM] in
 REPEAT STRIP_TAC THEN IMP_RES_TAC (snd (EQ_IMP_RULE th1)) THEN
    IMP_RES_TAC (SUBS sublist (snd (EQ_IMP_RULE th2))) THEN
    IMP_RES_TAC (SPECL ["m+n";"p+n";"p+q"] LESS_TRANS)
    THEN ASM_REWRITE_TAC []
```

Translation:

```
(implies (and (lessp m p) (lessp n q)) (lessp (plus m n) (plus p q)))
```

However, translating specifications appropriately for such systems can be difficult, often leading to an excess of axioms that can slow down the provers to the point where fully automatic proofs become impossible in practice.

We feel that what is needed is some kind of compromise between the two types of tools. To achieve such a compromise, we are developing PM as not simply an improved HOL interface, but as a general proof management interface that allows the user a choice of provers, and even allows different

provers to be used during the course a single proof. We note that although the experience of some human verifiers (e.g., [Win89]) leads us to believe that a choice of theorem provers will be very useful, there is a potential alternative benefit to a unified tool such as PM: it could provide for a standard theorem prover interface language that could allow designers of development and verification systems that incorporate a theorem proving capability support to factor out the question of which actual theorem prover to incorporate.

PM is based on the tree editor Tree-mode [Kamin90] [HKC85], an extension to GNU emacs, and is designed to maintain proofs in a tree structure in such a manner that the assertions in the children of a validated node are always guaranteed to imply the assertion in the parent. Nodes in a tree maintained by PM are validated via calls to a theorem-prover. It is this device which makes it possible to use different provers at different stages of a large proof. Once sound translation schemes from a common assertion language to different provers are in place, the validity of proofs obtained using PM depends only on the soundness of the theorem provers used to validate nodes.

Our initial PM prototype uses, as indicated above, the HOL assertion language as the common assertion language, and has first-order translation capabilities to provide interfaces to provers whose assertion languages are first-order. These include the Greenbaum resolution prover [Green86] and, thanks to Kaufmann's Skolemizer [Kau89], the Boyer-Moore prover. In its current form, one can assure that proofs obtained using this prototype simply as a HOL interface are valid. We cannot yet provide this assurance for proofs using additional provers: in the interest of testing the feasibility and utility of a multiple-prover interface, we are concentrating initially on implementing plausible translation schemes. Ultimately, we will justify their soundness in detail.

We do not underestimate the problems involved in the PM project. Most prominently, there are many questions to be answered about the translations to be provided between provers, and proofs (as opposed to plausible arguments) of translation soundness must be provided. In addition, we must determine the best way to maintain parallel versions of theories, and how to handle type information, abstract type definitions, recursive function definitions, and so on.

We do not have answers to all of these questions yet, but we have made a beginning. After discussing related efforts in section 2, we describe some of the problems inherent in the development of PM in section 3, and in section 4 report on what we have so far accomplished and on where we are concentrating our current efforts. In section 5, we discuss our future plans for PM. An appendix includes an example of what PM looks like.

2. Related work

Tree-mode, the tree editor underlying PM, comes from a line of tree-oriented editors and tree-based theorem-proving interfaces [HKC85, Ham88, Kamin90, Swarup88] developed at the University of Illinois, including interfaces to Nuprl [Swarup88] and the Greenbaum prover [HKC85]. None of these provided a true multiple-prover interface.

The current command-line interface to HOL [Kal91] is is an improvement over previous command-line interfaces in that HOL keeps track of the tree structure of the proof, and a user can attack the proof at any point in the tree. This provides a lot of flexibility in proof management. However, PM provides a HOL user with the additional advantages of direct and visual control of the proof. The process of moving the cursor in the tree to prove or edit components of the proof in a random access manner gives PM a great deal of extra flexibility. The ability to display the proof tree in many different formats also gives the user quick access to information about the proof not available in any other way.

Adding external theorem provers to a HOL environment is not the unique way to add heuristic proof to HOL. There are concurrent efforts to integrate such proof into HOL directly. There are attempts to implement features in HOL similar to the tactic AUTO-TAC in the Nuprl system which solves simple goals automatically. More sophisticated specification tools [Win90] can, by providing constructs such as abstract theories, create domain-specific auto-tactics. However, such auto-tactics are seldom as general or

efficient when applied to the same problems as automatic provers specialized to the same purpose.

Simplifying the process of proving essentially-first-order assertions by supplementing HOL has been done by the indirect use of FAUST [SKK91], a prover that is implemented in ML, and thus easy to integrate into the HOL environment. FAUST is a full first-order automatic prover based on natural deduction using a sequent calculus whose inference rules can be tranlated into HOL inference rules; thus, a FAUST proof essentially can be translated into a HOL tactic. The problem of equivalence of formulae and inference rules in FAUST and HOL is currently avoided by using FAUST to find a HOL tactic, and then running that tactic in HOL. Thus, the proof security provided by HOL is retained, although full advantage of the speed-up in the proof process is not yet being taken. It should not be difficult to include an interface to FAUST in PM.

3. The development of PM

In this section, we discuss in more detail some of the problems that arise in connection with the development of a combined theorem prover interface, and how we are handling or planning to handle them.

As will be elaborated in the next section, PM maintains proof trees containing the lemma structure that supports the proofs of the assertions at their roots. PM is currently primarily a HOL interface, and as such, maintains a theory node that contains the currently saved axioms, definitions and theorems available at each call to the HOL prover in the current proof trees. This information needs to be made available, so far as possible, to any other provers supported by PM that may be invoked in the course of a proof. In the cases we have examined so far, when this information is not higher-order, there exist translations suitable for the Boyer-Moore and Greenbaum provers which is of varying degree of directness. For example, recursive function definitions (e.g. [Ploegaarts91, Andersen91]) typically (modulo type information) translate easily into Boyer-Moore input, but require a (usually straightforward) formulation as a set of axioms for the Greenbaum resolution prover. First order formulae, on the other hand, can (again, modulo type information) be passed easily to the Greenbaum prover, but would need to use Kaufmann's Skolemizer or some similar device to be formulated for the Boyer-Moore prover. In at least some cases, second-order formulae can be expressed as first-order; this is discussed in more detail in the next section. The simple abstract and recursive type definitions we have examined so far also appear to have relatively easily obtained translations. Developing actual translation algorithms and discovering their limitations will be our next step. As discussed in the next section, we have a preliminary algorithm that handles a subset of HOL formulae.

So far in this section, in discussing translation, we have ignored the problem of type information. In typeless logics such as that of the Boyer-Moore and Greenbaum provers, this can be handled by using appropriate predicates. By the use of predicates, one can even express information about function types and polymorphic types. It is known that, in resolution proving, a more efficient way to handle types is to tag expressions of a given type with the type name, to prevent terms of different type from unifying. This device becomes much more difficult to use in the presence of function types and polymorphic types; it remains to be settled which scheme to implement for resolution proving, or whether to provide the user with a choice. Preliminary experiments have shown that a type tag scheme could be useful even in the presence of polymorphism and quantification of functions. [Archer92]

Other details to be worked out involve proof tree management simultaneously for provers that favor forward and backward proof, and how to handle sequents (the standard form for assertions in HOL) in provers that do not use them. However, there seem to be very natural solutions to these problems; for example, sequents could be handled by the creation of temporary children of a node, with a call to a non-sequent-style prover to show that the node assertion follows from the child assertions introduced as temporary axioms.

4. The current status of PM

As an interface to HOL, PM is well-developed. Facilities exist for moving around in the tree, adding and removing nodes, copying subtrees to new locations, calling the HOL system with a given tactic to be applied to the goal stored at a node, and compacting contiguous sets of validated nodes and their tactics into a single node and tactic. The result of applying a tactic to a goal is a (possibly empty) set of subgoals which are guaranteed by the tactic to imply the original goal; for valid tactics, the corresponding subgoal nodes are automatically added to the goal node as its children; some of these operations cause the status of a node (which indicates whether or not it is valid and how it is known to be so) to be updated appropriately. There is also a capability of maintaining function and tactic definition and type information in conjunction with the whole tree.

When a valid tactic is applied to a node, that tactic is saved as part of the information at that node (displayed only by request), so that the information maintained in a tree, all of whose nodes have been completely validated, will be sufficient to construct a proof of the root. Maintaining such information also allows one to use information stored at a node to re-prove the node from its children; this can, for instance, permit one user to confirm the validity of a proof constructed by another. Until the soundness of all tree-editing operations is formally proved, this ability to re-prove is the guarantee of soundness of HOL proofs as documented by PM (although a result proved using HOL from PM is guaranteed to be as valid as one proved using HOL directly). PM proofs using only HOL can also be checked by taking advantage of the compaction feature of PM; applied to a complete HOL-PM proof tree for an initial goal, compaction will produce a pure HOL proof which can be confirmed using HOL directly. Currently, this is the method provided for assuring HOL proofs in PM to be solid. When soundness proofs for PM are completed, such extra assurance will not be necessary.

In a proof tree maintained by PM as a multiple theorem prover interface, nodes are ''validated'' by calls to an external prover. The information at a validated node (in the case of HOL, a tactic and type information) includes a record of how to reproduce the proof that the goal at the node follows from the subgoals at the node's children.

HOL proof construction in PM typically proceeds in a backwards fashion, by applying tactics and tacticals to the goal, attempting to reduce the goal to simple axioms and theorems already proved. Theorem provers such as the Boyer-Moore prover and the Greenbaum prover support a lemma-driven, forward proof style in which one first derives (or proposes) lemmas from which the theorem prover can derive the goal. Using forward proof in connection with the maintenance of a proof tree involves permitting the ad hoc—rather than automatic—addition of child nodes to a node. The details of how best to manage combined support for both styles remain to be settled.

As we have indicated, a rudimentary capability for passing a subset of HOL formulae to the Greenbaum prover is now in place. With simple adjustments for syntax, this will permit us to extend this capability to the Boyer-Moore prover, thanks to Kaufmann's Skolemizer.

The subset of HOL formulae that can be appropriately translated includes all that are manifestly first-order. However, additional formulae can sometimes be handled in a natural way by translating quantified variables of function type as undefined function symbols, or in some cases of boolean-valued functions, undefined predicate symbols. Other apparently higher-order assertions, for example, ones involving equality of quantified formulae, can be handled by translating equality as the logical operator iff. Currently, our algorithm works as a filter, first determining whether it is able to produce a translation.

An interesting feature of our first-order extraction algorithm is its analysis technique for determining when to translate potential logical operators as meta-level logical operators, and when to translate them as low-level operations in some theory of booleans. There is sometimes a choice, but for some provers, including resolution provers, it is usually more efficient to use meta-level operators when possible (unless their use requires extra axioms to connect them semantically to the low-level operators).

Here are some example translations that our algorithm provides. The first example illustrates the mixture of meta-level and low-level boolean operators. ''And'', ''Or'', ''D'', and are four HOL

definitions used in the goal. These definitions are used as rewrites in HOL to produce the rewritten goal, which is then translated into input for the Greenbaum resolution prover. The translated assertion and definitions shown below have a lisp-like format, in which "A" and "E" stand for "for all" and "there exists" respectively.

Original:

And = $\vdash \forall$ a b c. And a b c = (c = a $\wedge$ b)
Or = $\vdash \forall$ a b c. Or a b c = (c = a $\vee$ b)
D = $\vdash \forall$ a b c d out. D a b c d out = (out = a $\wedge$ b $\vee$ c $\wedge$ d)
D_imp = $\vdash \forall$ a b c d out. D_imp a b c d out = ($\exists$ p q. And a b p $\wedge$ And c d q $\wedge$ Or p q out)
goal = $\forall$ a b c d out. D_imp a b c d out = D a b c d out

Rewritten Original:

$\forall$ a b c d out. ($\exists$ p q. (p = a $\wedge$ b) $\wedge$ (q = c $\wedge$ d) $\wedge$ (out = p $\vee$ q)) = (out = a $\wedge$ b $\vee$ c $\wedge$ d)

Translation:

```
(A (a b c d out)
  (IFF (E (p q) (AND (= p (and a b))
                     (= q (and c d))
                     (= out (or p q))))
       (= out (or (and a b) (and c d)))))
```

The second illustrates the introduction of undefined predicate symbols for quantified booleans. Note that "a" and "b" below become undefined 0-ary predicates, and "f" a unary predicate.

Original:

$\forall$ a f n. a $\wedge$ f(n) $\vee$ f(n) $\wedge$ b = f(n) $\wedge$ (a $\vee$ b)

Translation:

```
(A (n) (IFF (OR (AND (a) (f n)) (AND (f n) (b)))
            (AND (f n) (OR (a) (b)))))
```

Both examples prove faster with the Greenbaum prover than with HOL, and do not require construction of a tactic.

One can go even farther than this in translating for first-order provers. For example, one can handle additional higher-order formulae by introducing an APPLY operation that becomes a new undefined function symbol, accompanied by appropriate axioms to translate the definition of function equality, function construction, and type information. The question here is how far it is worth going. As soon as one has to add axioms, one increases the size of the search space of an automatic prover. The translation can start to become cumbersome and unnatural. Nevertheless, our inclination is to provide, eventually, as extensive a translation capability as possible, and then leave it up to the user of the system whether it is worth employing it in a particular case.

5. Ongoing and future work

We see PM as not only a tool for creating proofs, but a tool for documenting proofs. For the latter purpose, the appropriate use of squashing contiguous subsets of nodes can produce a tree with the appropriate conceptual proof structure. While we provide for the simultaneous compaction of tactics, the soundness of this compaction in all cases remains to be formally proved. Currently, to be totally confident of the correctness of individual compacted HOL tactics, one can rerun them in HOL. However, since only groups of validated nodes can be squashed, it is never in question that a parent assertion follows from its child assertions.

In general, we expect to store proof trees in such a form that, assuming all their nodes have been validated, the entire proof (or specific complex sub-proofs) can be automatically rerun on request. That

is, a completely validated tree will effectively constitute a proof script.

As we have indicated in the introduction, our initial aim is to investigate the feasibility of combining existing provers in a multiple-prover tool and the application of several provers in the course of a single proof. In addition to work on translation schemes, this effort will include applying the prototype extensively to real verification examples.

Detailed proofs of soundness will eventually be undertaken, both for translation schemes and for proof compaction schemes and other tree-editing operations.

We are beginning with just three provers: HOL, the Boyer-Moore prover, and the Greenbaum resolution prover. Each additional prover incorporated into PM will require some effort in developing a corresponding translation scheme and translation soundness proof, although the general first-order translation algorithm will help to factor this effort. It should not be especially difficult to incorporate FAUST, for example. In addition, we plan to add an interface to a BDD tool that handles binary decision diagrams, and develop other translation schemes to interface with various useful decision procedures.

Beyond creating fixed translation algorithms, other translation possibilities arise. We note that there can sometimes be a choice of how to translate a given formula; for example, in the decision of which boolean operators should be treated as low-level or meta-level. There are other cases in which generalization, e.g., in the form of ignoring type information, may be useful. In such cases, we may provide the user multiple options besides the default. A further ambition for the future is to take advantage of the availability of alternate provers by developing techniques for recognizing when sequents and formulae or sub-formulae are most suitable for handling by a particular prover.

References

[A-FL91] James Alves-Foss and Karl Levitt, *Verification of secure distributed systems in higher order logic: a modular approach using generic components*, Proceedings of the 1991 IEEE Symposium on Security and Privacy, May 1991.

[Andersen91] F. Andersen and K.D. Petersen, *Recursive Boolean Functions in HOL*, to appear in Proceedings of the 1991 International Tutorial and Workshop on the HOL Theorem Proving System and its Applications, August, 1991.

[Archer91] Myla Archer, George Fink, and Yang Lie, *Linking theorem provers to HOL using PM: Proof Manager*, submitted to the 1992 International Workshop on the HOL Theorem Proving System and its Applications.

[BM88] Robert S. Boyer and J Strother Moore, A Computational Logic Handbook, Academic Press (1988).

[Cohn88] A. Cohn, *Correctness properties of the Viper block model: the second level*, Technical Report 134, Computation Laboratory, University of Cambrigde, 1988.

[Gordon87] Mike Gordon, *HOL: A proof generating system for higher-order logic*, in: VLSI Specification, Verification and Synthesis (G. Birtwistle and P. A. Subrahmanyam, eds.), Kluwer (1987).

[Green86] Steven Greenbaum, *Input Transformations and Resolution Implementation Techniques for Theorem Proving in First Order Logic*, Doctoral Dissertation, University of Illinois at Urbana-Champaign, 1986 (also TR UIUCDCS-R-86-1298, Department of Computer Science, University of Illinois at Urbana-Champaign).

[HKC85] David Hammerslag, Samuel N. Kamin, and Roy H. Campbell, *Tree-oriented interactive processing with an application to theorem-proving*, Proc. of the ACM/IEEE Conf. on Software Development, Tools, Techniques, and Alternatives (SoftFair II) (December, 1985).

[Ham88] David Hammerslag, *Treemacs Manual*, TR UIUCDCS-R-88-1427, Department of Computer Science, University of Illinois at Urbana-Champaign, 1988.

[Joyce88] J. Joyce, *Formal verification and implementation of a microprocessor*, in: G. Birtwistle and P. Subrahmanyam, eds., VLSI Specification, Verification and Synthesis, Kluwer Academic Publishers, 1988.

[Kal91] Sara Kalvala, *Building Interfaces to HOL*, to appear in Proceedings of the 1991 International Tutorial and Workshop on the HOL Theorem Proving System and its Applications, August, 1991.

[Kamin90] S. Kamin, *Differences between Treemacs and Tree-mode*, University of Illinois at Urbana-Champaign, 1990.

[Kau89] Matt Kaufmann, *An Extension to the Boyer-Moore Theorem Prover to Support First-Order Quantification*, to appear in Journal of Automated Reasoning, also, CLI Technical Report 43, May 1989, Computational Logic, Inc., Austin, Texas.

[Ploegaarts91] W. Ploegaarts, L. Claesen, and H. De Man, *Defining Recursive Functions in HOL*, to appear in Proceedings of the 1991 International Tutorial and Workshop on the HOL Theorem Proving System and its Applications, August, 1991.

[SKK91] Klaus Schneider, Ramayya Kumar, and Thomas Kropf, *Integrating a First-Order Automatic Prover in the HOL Environment*, to appear in Proceedings of the 1991 International Tutorial and Workshop on the HOL Theorem Proving System and its Applications, August, 1991.

[SL91] E. Thomas Schubert and Karl N. Levitt, *Verification of Memory Management Units*, 2nd IFIP Working Conference on Dependable Computing for Critical Applications, February 1991.

[Swarup88] Vipin Swarup, The UIPRL Proof Development System Master's Thesis, University of Illinois at Urbana (1988).

[Win89] Philip J. Windley, *Private communication*, 1989.

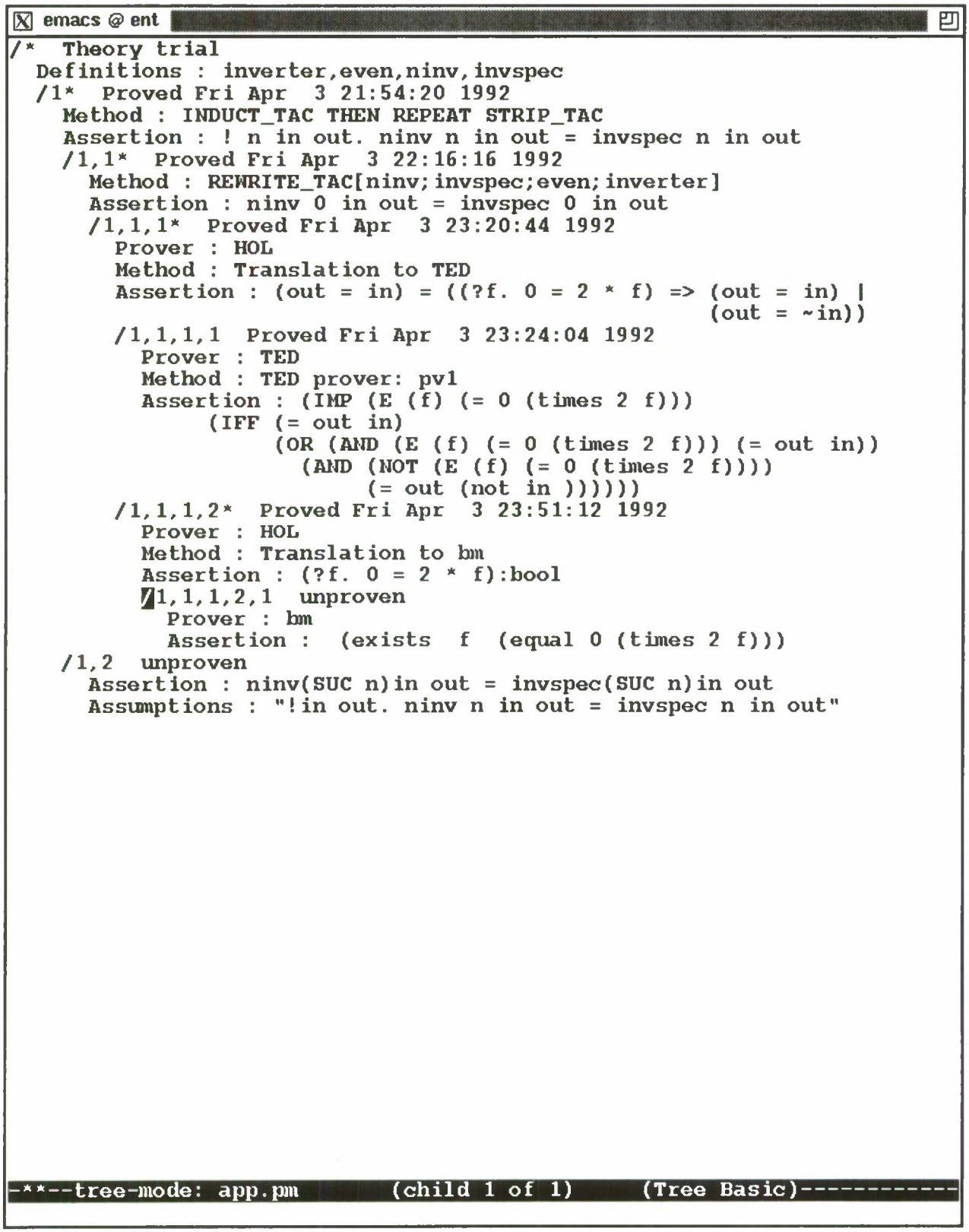
[Win90] Philip J. Windley, *The Formal Verification of Generic Interpreters*, Ph.D. thesis, University of California, Davis, 1990.

[Win91] Phillip Windley, *Abstract Hardware*, Proceedings of the 1991 International Workshop on Formal Methods in VLSI Design, January, 1991.

Appendix - PM Example

To give a flavor of the use of PM, a sample screen of PM is shown at the end of the paper. This displays a proof in progress; the root node of the tree contains the definitions of the theory, and the only child of the root (labeled "/1*") is the node containing the assertion to be proved. As in the HOL notation, "!" stands for ∀ and "?" for ∃. Two subgoals of the proof have been translated to the TED and Boyer-Moore provers.

PM can be obtained by contacting the corresponding author or on anonymous ftp at ted.cs.uidaho.edu in the file /pub/hol/pm.tar.Z.


emacs @ ent
/* Theory trial
Definitions : inverter,even,ninv,invspec
/1* Proved Fri Apr 3 21:54:20 1992
Method : INDUCT_TAC THEN REPEAT STRIP_TAC
Assertion : ! n in out. ninv n in out = invspec n in out
/1,1* Proved Fri Apr 3 22:16:16 1992
Method : REWRITE_TAC[ninv;invspec;even;inverter]
Assertion : ninv 0 in out = invspec 0 in out
/1,1,1* Proved Fri Apr 3 23:20:44 1992
Prover : HOL
Method : Translation to TED
Assertion : (out = in) = ((?f. 0 = 2 * f) => (out = in) |
(out = ~in))
/1,1,1,1 Proved Fri Apr 3 23:24:04 1992
Prover : TED
Method : TED prover: pv1
Assertion : (IMP (E (f) (= 0 (times 2 f)))
(IFF (= out in)
(OR (AND (E (f) (= 0 (times 2 f))) (= out in))
(AND (NOT (E (f) (= 0 (times 2 f))))
(= out (not in))))))
/1,1,1,2* Proved Fri Apr 3 23:51:12 1992
Prover : HOL
Method : Translation to bm
Assertion : (?f. 0 = 2 * f):bool
/1,1,1,2,1 unproven
Prover : bm
Assertion : (exists f (equal 0 (times 2 f)))
/1,2 unproven
Assertion : ninv(SUC n)in out = invspec(SUC n)in out
Assumptions : "!in out. ninv n in out = invspec n in out"
-**--tree-mode: app.pm (child 1 of 1) (Tree Basic)----------

Potential Benefits From Implementing the Clark-Wilson Integrity Model Using an Object-Oriented Approach

by
Craig A. Schiller
Science Applications International Corporation
30 June 1992

Prepared for the 1992
National Computer Security Conference

ABSTRACT

In 1987, Dr. David Clark and Dr. David Wilson introduced the Clark-Wilson Integrity model to the world. Their landmark paper [CLARK87] points out that commercial systems require integrity more than they need confidentiality. The paper goes on to describe a model for achieving and preserving integrity.

For the past 20 years or so, a paradigm for creating systems based on objects instead of processes, has evolved. The "object-oriented" paradigm claims that systems created in this manner result in better models of the real world entities that they represent.

This paper describes potential benefits from a system development approach that uses object-oriented concepts while implementing the Clark-Wilson Integrity model. The paper documenting this approach, "An Object-Oriented Strategy for Implementing the Clark-Wilson Integrity Model", was presented at the 1992 Information Systems Security Association (ISSA) conference in Houston, Texas (Copies will be made available at the session).

POTENTIAL BENEFITS FROM IMPLEMENTING THE CLARK-WILSON INTEGRITY MODEL USING AN OBJECT-ORIENTED APPROACH

by
Craig A. Schiller
Science Applications International Corporation
600 Gemini MS# R11A
Houston, TX 77058

1. Introduction

This paper describes the potential benefits from a system development approach that uses the Clark-Wilson Integrity model and Object-Oriented methods and concepts. The benefits described are a result of the software engineering practices, the use of Ada, the Object-Oriented methods and concepts, and the Clark-Wilson Integrity model presented in [12]. The intended audience includes individuals interested in high integrity systems, concepts for addressing integrity issues, object oriented technology, or the Clark-Wilson Integrity model, and vendors of security products and services. Due to space limitations, the reader is assumed to have a basic understanding of both the Clark-Wilson Integrity Model and the Object-Oriented Paradigm.

The purpose of this paper is to:

- Stimulate discussion
- Explore cross-fertilization benefits resulting from the combination of software engineering and security engineering
- Present findings from a proposed implementation of the Clark-Wilson Integrity model
- Share insights gained as a result of researching the application of object-oriented concepts to the Clark-Wilson Integrity model.

1.1 Observations

By themselves, traditional security controls (e.g. access.controls) cannot ensure integrity. As you would expect, many of the controls, to achieve and preserve integrity, operate at the application software level. Integrity is, primarily, an application developer and user responsibility. Traditional security controls can preserve integrity, once acheived, but to meet a desired integrity-state, or change integrity-values requires the user/developers data specific knowledge. To achieve and preserve integrity a developer will determine the integrity requirements, then design and develop a system to meet these requirements. Controls must exist throughout definition, design, development, testing, operations, and maintenance of a system. The Clark-Wilson model assists developers by identifying strategic and application essential functions, relations, and processes affecting integrity. The approach, described in the companion paper "An Object-Oriented Strategy for Implementing the Clark-Wilson Integrity Model" [12], discusses an implementation of the Clark-Wilson model using existing technology.

Integrity is a complicated subject. Every new insight is accompanied by a myr-

iad of new questions. A single, simple answer for integrity issues does not exist. Instead, a process is needed that will assist the developer, maintainer, and user in adequately covering integrity concerns. There are indications that the object-oriented paradigm and the Clark-Wilson Integrity model may produce synergistic benefits from their combined use.

1.2 Organization of the Paper

This paper will describe the potential benefits from a hypothetical set of integrated application and system controls intended to achieve and preserve high levels of integrity. Note that not all objects in a system merit these levels of integrity controls. One of the early steps in the development process is to determine which objects require extended Integrity/Availability measures.

First the Clark-Wilson model will be described in terms of objects. A summary of the implementation steps of the object-oriented strategy will be provided. The benefits from an object-oriented implementation of the Clark-Wilson model will be followed by recommendations for vendor support of this approach. A few final comments will conclude the paper.

2. Applying Objects to the Clark-Wilson Model

Figure 1 illustrates the Clark-Wilson Integrity model from an object oriented perspective. As stated earlier, the intent of this paper is to show how an object-oriented approach might be used in an implementation of the Clark-Wilson Integrity model. Clark-Wilson and object-oriented concepts mutually benefit from an implementation approach based on both perspectives. In other words, the result of the joining of these two concepts is greater than each would have achieved independently.

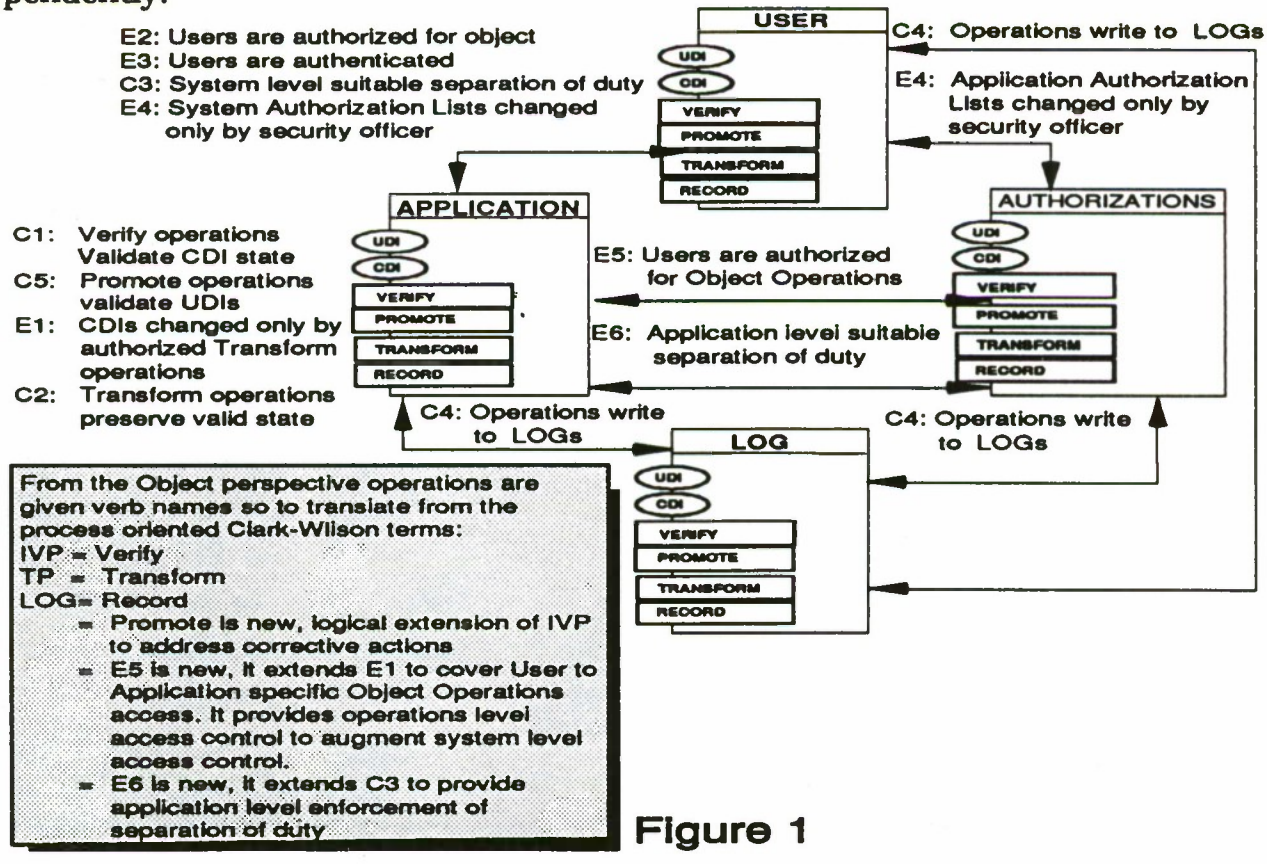


Figure 1

When the Clark-Wilson model is described in Object-Oriented terms four major objects emerge. They are:

A USER object	
An AUTHORIZATIONS object -	to support the Access Control Triple and the Separation of Duty concept. Application objects request authorization to honor an operation request. The Authorizations object uses authorization data, state data, and knowledge of relationships to grant authorization.
A LOG object -	To support the Well-Formed-Transaction
Application objects -	One or more

Each object can contain both unconstrained and constrained forms of data. The unconstrained forms are used when data is input (e.g. translated from ASCII to a typed variable), received from a lower integrity system, or received from an external source. The constrained forms are used for modeling the real world or logical entity. These are the strongly typed variables that are used for calculations and object values.

Each object will contain four types of operations:

- Verify - The equivalent of the automated portion of the Clark-Wilson Integrity Verification Process (IVP), translated into a verb. In the object-oriented paradigm, operations are given verb names.
- Promote - A special case of the Clark-Wilson Transformation Process (TP). Used in transitioning data from external sources or from systems of lower integrity.
- Transform - The equivalent of the Clark-Wilson TP. Only the allowed object operations are visible to other objects.
- Record - The equivalent of the Clark-Wilson LOG. This includes the different types of logs as identified in the follow-on NIST workshops [4], [8], [9]

In an actual implementation, the concepts presented are to be applied in varying degrees depending on the integrity and availability requirements of each object. Ideally, the objects will be grouped into processing domains according to the nature of integrity controls required. More than likely, each object's integrity requirements will vary. The degree of this variation will probably dictate that objects of similar integrity controls and limits will be grouped into domains. In writing this paper an assumption is made that each domain will exist on different processing nodes in a distributed system, each node running operating systems with only TCSEC (Trusted Computer System Evaluation Criteria or Orange book)[13] C2 level capabilities. However, as operating systems continue to evolve, it may be possible to have the different domains coexist in a single, affordable platform.

3. Summary of Proposed Implementation Steps

Due to space limitations, Tables 1 and 2 summarize the proposed implementation strategy. The detailed implementation steps can be found in [12]. These steps identify tasks for developing a system incorporating the Clark-Wilson Integrity model and the Object-Oriented concepts.

Requirements and Development Phase

I1. During requirements definition, explicitly declare the integrity, availability, confidentiality, and measurability requirements for each object. Identify objects requireing extended integrity measures.

I2. During development, include, in the object specification, only those operations that are intended for use by other objects and users,

I3. During development, include at the beginning of each object operation procedure (in the object body), a request to the Authorizations object that it perform the following:

1. Verify that the object's state, system's state, user's state, and other related object's states permit a requested operation.

2. Verify that an individual (or an individual's position) is permitted to request a specified operation.

I4. During the requirement and development phases,

1. Perform a deliverable-specific, life cycle phased risk assessment of the integrity, availability, confidentiality, and measurability requirements of the system and its objects.

2. Perform certification testing of these requirements as part of system testing.

3. Bond the final system product using a non-repudible authentication technique.

Table 1

4 Potential Benefits and Concerns From the Research

The potential benefits listed below are benefits of the approach described in [12]. As such, some of the benefits described are a product of aspects of the approach other than object-oriented concepts and the Clark-Wilson model. These other aspects include the use of good software engineering practices; the use of digital signature for quality assurance, version control, and authentication; etc.

4.1 Potential Benefits

a. **Objects can provide a close correlation to their real-world counterparts**

The closer correlation makes it easier to communicate with users of the system. Developers can discuss issues or concerns in real-world terms vice programming terms, resulting in a better model. It should also be easier to confirm that the logical state (the state of the real-world entity reported by the object) reflects the real-world state (the actual state of the real-world entity). This is turn makes it easier to verify the integrity-state of the objects. This close correlation between the real-world and the object-world provides a smooth transition from real-world concerns to high level logical concerns. The smooth transition through each layer of abstraction, down to the lowest layer of detail, is a primary benefit of the object-oriented approach. This characteristic of the object-oriented paradigm enhances the ability of the Clark-Wilson model to verify correspondence with the real-world.

b. **Strong typing, related error detection, and error handling contributes to better real-world modeling and support for the Well-Formed-Transaction.**

The strong typing, error detection, and error handling features of some structured, object-based languages support better real-world models by providing the mechanism for describing the logical object in terms of real-world limits. In doing so, these languages relieve the programmer of having to explicitly code many limit checks and reduces the overall complexity of the programming task.

The error detection and error handling features of some object-based languages trap errors when they occur but allow them to be handled at a developer specified level. The developer can develop a strategic error handling hierarchy that will be enforced by the application.

For example, the system designer may categorize data into the following groups:

- Data whose errors are to be handled on a field by field basis as they occur
- Data whose errors are to be collected and handled on an object basis
- Data whose errors are to be collected and handled at the application level.

In addition, non-data errors can also be grouped according to a hi-

Operations Phase

I5. During the operations phase,

1. Authenticate that the system being executed is the bonded system;
2. Authenticate that all components are the correct version from the official source.
3. Authenticate that the system has not been modified since bonding;
4. Certify and accredit the system for operational use;
5. Certify and accredit the system for use with its external interfaces.

I6. During certification and accreditation (steps I5.4-I5.5) certify that:

1. The static data and code is unchanged from when the data was bonded;
2. The states of the object, system, user, and related objects are valid (using IVPs);
3. The object state and data correspond to the real-world entity they represent (using IVPs).
4. The system meets its inte3grity, availability, and confidentiality requirements (from the certifying agents' perspective).

I7. Enforce a strict integrity policy. Use isolation, IVPs and a promotion procedure for promoting data from a low integrity domain to a higher integrity domain.

I8. Use discretionary access controls to deny access to data except through its object. If supported, deny execute privilege to all users except those that need to execute object operations. Use extended user-authentication to identify and verify users to the system.

I9. Maintain lists of relations (using the Authorizations object and system security software) between users, operations, states, and objects.

Table 2

erarchical error handling scheme. Errors might be handled at the module, object or application level.

Managing the handling of errors also assists the developer in implementing Well-Formed-Transactions. When an error is handled the developer can:

1 Flag the error
2 Accept the error
3 Accumulate error statistics
4 Correct the error with an estimate (e.g. previous value or trend extrapolation)
5 Correct the error with a reconstructed value
6 Correct the error with a default value
7 Correct the error with a request for replacement from the source
8 Reject the data and restore the data (from the log) to the original value that existed before the transaction began).
9. Reject the data and continue processing.

The developer makes a choice based on the nature of the system and data. The choice also depends on the robustness of the design of components that use this data.

c. **The encapsulation of data and processes into objects enables the formation of the Clark-Wilson access control triple.**

Objects automatically encapsulate the data and related operations. By binding the user to an object (authorizing a user to use an object) and a set of it's operations, the access control triple is achieved. Concerns about the implementation of the access control triple are addressed in the following sections.

d. **Information hiding reduces the complexity of the system for developers**

The object developer reduces the complexity (for other developers) of the object by making visible only those operations which are intended for external use. Hiding those processes that are not intended for external use will reduce the chance that a developer will use an internal process by mistake. There is less chance that a developer will incorrectly sequence processes if the visible process controls the sequencing of internal steps. Information hiding reduces the number of procedures to be learned to invoke a processes. Information hiding also supports the concept of the access control triple by reducing the number of procedures that have to be controlled with an access control triple.

e. **The Authorizations Object is a key component of the implementation approach.**

The Authorizations Object provides enforcement of a strict integrity policy, even if the concept is not supported by the native operating system. The Authorizations Object is called each time an operation on an object is requested. The Authorizations Object verifies that the requesting object is of equal or lower integrity than the source object. When this is not true, either rejection or promotion must occur.

The Authorizations Object provides a mechanism for enforcing role and user based authorizations. This mechanism forms the basis for an automated enforcement of separation of duty restrictions. Separation of duty restrictions reduce the risk of fraud via privilege abuse and increase integrity via n-person control of critical operations. This mechanism also forms the basis for the Access Control Triple enforcement mechanism. The Authorizations Object restricts access to and operations on constrained data items. Only developer supplied (or identified) operations are authorized for use on objects. These restrictions will reduce the risk of error by extending the concept of information hiding to a finer level of granularity.

The Authorizations Object automatically enforces integrity, availability, confidentiality and related state restrictions. It allows the developer to specify object and operation specific responses to integrity/availability fault or failure. The Authorization Object can be used to automate the invocation of the Integrity Verification Procedure (IVP).

f. **Objects and object instantiation can provide support for the formation of integrity and isolation domains[2].**

The Authorizations Object can be used to automate the promotion process required between domains of different integrity. This will reduce the possibility of corruption caused by the blind exchange of data with unknown integrity. In the approach, an object is instantiated in the isolation domain to allow the promotion to take place without endangering critical objects.

g. **Applying risk management techniques on objects to assure Clark-Wilson integrity/availability concerns contributes to the certification of transformation processes.**

The Clark-Wilson model states that the Transformation Process must be certified but does not describe the nature of the certification required. In the proposed approach, risk management is used at an object level to provide visibility into the integrity , availability, and confidentiality design decisions. This improves the chances that design errors will be detected and trade-off decisions will be reviewed. The accepted residual risk is documented and is then reviewable as a result of this approach. Certification testing is used to confirm the success of the above assurances. The system should be bonded with a non-repudible authentication technique to guarantee the certification survives the transition from development to operations. This form of bonding ensures that the use of certified and accredited processes can be confirmed. In addition, bonding can be extended to ensure that only the correct version of software or data modules may be executed.

---

[2] An integrity domain is a grouping of objects with similar integrity requirements that are protected from influence by objects outside of the domain. An isolation domain is a domain that may be isolated when performing operations on data of uncertain integrity. A domain (in this paper) is a protected or bounded processing environment. For this approach, it has been specified that each domain exists on a separate platform or processing node, recognizing that more capable operating systems are able to maintain multiple domains within one platform.

4.2 Concerns

a. The Authorizations Object is central to the success of this approach. The approach can only be successful if the privilege grants are designed and implemented well by both the developer/maintainer and the administrator.

The Developer/Maintainer is responsible for:

Researching the affect of states on each operation

Identifying the capabilities required to support desired roles

Identifying those capabilities that are needed by all (unrestricted)

Identifying non-role specific restricted capabilities

Identifying separation of duty required restrictions

Determining the desired level of granularity of operations control

Determining which objects should be CDIs.

The Administrator is responsible for:

Honoring access requests form authorized owners, custodians, and stewards.

Responding in a timely manner to requests

Documenting and reporting needed modifications to authorizations object logic or data.

b. The Clark-Wilson model can be subverted if DBMS or flat file editing is able to modify data without using the certified object operations.

c. Finally, and most important,

Management must be certain that enough is known about the application to provide for every need before applying the Clark-Wilson Integrity model to an object.

If the Clark-Wilson model is implemented on data that is not well understood, then the user may not be able to perform vital functions using the developer supplied operations. Unfortunately, the user will also not be able to get to the data using other, less restrictive means. For this reason, even if the data is very well understood, a plan should be developed to accommodate emergency access to the data should it be required. Just as a facility prepares contingency plans in case of accident, so should the developer prepare contingencies for unanticipated emergencies.

5. Conclusions

5.1 Recommendations for vendor Software support

Five product enhancements by vendors supplying operating systems, security software, and data base management systems will make this implementation easier.

a. Provide an Authorizations Object service that applications may use to augment system discretionary access control with application specific access controls such as operations level access control and application level separation of duty enforcement.

b. Provide a Log Object service that applications may use to implement the Well-Formed-Transaction concept.

c. Provide an easy mechanism for application developers to bind object operations to DBMS data and to preclude all other access to designated data.

d. Provide a digital signature engine as a system service.

e. Provide linkage that employs the digital signature technology as an executive management function. As part of the system build and executable load process this would be useful to configuration management and quality assurance.

5.2 Summary

In conclusion, some of the issues raised by this research are more complicated than the original query (Levels of integrity, promotion, etc.). Some of the side issues have far reaching impact (binary/analog nature of integrity, guarantees of valid certification, etc.). Clearly, there is no simple answer. A process is needed for providing integrity. I believe that the Clark-Wilson Integrity Model and the Object-Oriented paradigm are two components of that process.

The Clark-Wilson Integrity model has provided focus and attention on the need for integrity. The Object-Oriented paradigm has much to offer toward that end. As in the companion paper, I hope that this paper encourages others to find ways to marry these two promising technologies with new and innovative ideas to improve the integrity and availability of our critical systems. There is much work to be done in the area of integrity engineering, and the future appears promising.

Acknowledgements

The author would like to thank Donald L. Evans (Paramax) for his hours spent helping me refine this paper, J. David Thompson (SAIC) and Ed Kusik (RSOC) for their invigorating technical discussions of new ideas, Ron Harris (Rockwell Space Operations Company) for coining the terms *integrity-state* and *integrity-value*, the members of the Mission Operations Directorate (NASA/Johnson Space Center) and the MOD AIS Security Engineering Team (ASET) for providing a fertile and productive environment for developing new ideas, and finally Susan Pernia (SAIC), Rich Owen (NASA), Steve Green (Paramax), Bob Smock (RSOC) and Jim Broadfoot(RSOC) for giving me the opportunity to work with NASA and the MOD ASET.

BIBLIOGRAPHY

1. Mission Operations Directorate AIS Security Engineering Team (MOD ASET) AIS Security Reference Structure JSC-25285 Projected publication date September 1992

2. Grady Booch. Software Components with Ada. The Benjamin/Cummings Publishing Company, Inc. 1987

3. Grady Booch. Software Engineering with Ada. The Benjamin/Cummings Publishing Company, Inc. 1987 (Second edition)

4. David D. Clark and David R. Wilson. "A Comparison of Commercial and Military Security Policies". In IEEE Symposium on Security and Privacy, pages 184-194, April 1987.

5. David D. Clark and David R. Wilson. "Evolution of a Model for Computer Integrity". In Proceedings of the 11th National computer Security Conference, October 1988

6. Bryan Flamig. Turbo C++ — A Self Teaching Guide. John Wiley and Sons, 1991

7. Harmon and Sawyer, Object Craft. Addison-Wesley Publications, 1991.

8. Stuart W. Katzke and Zella G. Ruthberg, Editors, Report of the Invitational Workshop on Integrity Policy in Computer Information Systems (WIPCIS), NIST Special Publication 500-168, January 1989.

9. Zella G. Ruthberg and William T. Polk, Editors, Report of the Invitational Workshop on Data Integrity, NIST Special Publication 500-168, September 1989

10. Thomas R. Malarkey, Integrity in Automated Information Systems, NCSC C Technical Report 79-91, September 1991.

11. D. T. Ross, J. B. Goodenough, C. A. Irvine. "Software Engineering Process, Principles, and Goals", Computer, May 1975.

12. Craig A. Schiller, "An Object-Oriented Strategy for Implementing the Clark-Wilson Integrity model.", Proceedings of the 9th Annual Conference for Information Security Professionals, sponsored by ISSA, 22-27 March 1992.

13. DoD Computer Security Center, DoD Trusted Computer System Evaluation Criteria, CSC-STD-001-83, 15 August 1983.

Precise Identification of Computer Viruses

Lawrence E. Bassham III
W. Timothy Polk

virus-lab@csmes.ncsl.nist.gov
National Institute of Standards and Technology
Computer Security Division

Abstract

The number of personal computer viruses continues to grow at an alarming rate. Many of these viruses are variants (i.e., close relatives) of "old" viruses. This often results in less than accurate identification of viruses. The consequences of this can be distressing: virus removal software fails, systems exhibit unexpected side effects, and researchers waste valuable time separating new copies of "old" viruses from new viruses. As a result, a public domain technique for precise identification of viruses is needed. This paper explores various alternatives.

Introduction

Why Precise Identification is Needed

The large number of variants of computer viruses complicate the processes of identification and eradication of viruses. Traditional virus identification methods may result in an incorrect, or at least imprecise, identification. This is a problem for users and researchers alike. Precise identification of computer viruses is required to obtain accurate information about the side effects of a virus, since such side effects can vary widely between variants. Accurate removal of viruses from infected executables also requires precise identification of the computer virus. Finally, researchers need precise identification techniques to separate new variants from known viruses efficiently to assist in the reduction of workload.

What is Meant by "The Same"

The first question to be addressed is the degree of precision required in this process. This may seem obvious but the relationship between samples is not always clear. Viruses which are self-encrypting or self-modifying may appear very different in different samples. Even simple viruses can include data areas that are variable.

There are many virus samples which are functionally equivalent. They may differ in the textual messages found in unused portions of the virus (e.g., various Stoned variants). Others may differ by replacing assembly language statements by their functional equivalents. Such a replacement might be the substitution of "Store zero in AX (MOV AX,0)" with "Exclusive Or registers AX and AX, storing the result in AX (XOR AX,AX)." These virus samples may also be removed by identical procedures. It is fair to ask if these viruses are the same or not.

Virus researchers do not agree whether these functionally equivalent viruses constitute "different" viruses. As a result, such basic questions as "How many viruses are there?" produce a bewildering range of answers. In late 1991, virus researchers and product developers were answering this question with estimates ranging from 350 to 1200.

In a heuristic sense, virus samples represent the "same" virus if both are descended from the same "base" virus without human modification. A more precise definition is desirable. First, a suitable definition of "virus" is required, though.

For the purposes of this document, the following definition of a computer virus will be used:

> A computer virus is executable code V which creates a functionally equivalent duplicate V' and binds the new copy to existing code P, creating P', in such a way that future execution of P' may cause execution of V'.

Note that V and V' must be functionally equivalent, but may differ as a bit-stream. This difference may be due to modification of data or addressing information, variable encryption, or other self-modifying techniques. In this case, P and P' are clearly infected with the same virus. More generally:

> P' and P'' are infected with the same virus V iff P' and P'' could have both evolved from some P infected by V without occurrence of a hardware error.

This definition has certain consequences. By this definition, the functionally equivalent virus samples described above are all different viruses. However, the self-modifying viruses may be very different samples, but are the same virus.

Theoretical Precise Identification

Theoretically, precise identification is achieved by performing a byte-for-byte comparison of the constant portions of a virus in an infected executable. In the case of self-garbling viruses, de-garbling must be accomplished before the comparison is performed. In practice, byte-for-byte comparison is impractical and undesirable. Byte-for-byte comparison would require the undesirable transmission of virus code in order to perform comparisons.

What is Actually Needed

An approximation of a byte-for-byte comparison process is required, allowing a user or researcher to determine the exact virus with an extremely high degree of accuracy without a library of virus samples. This can be accomplished with an accurate profile, or map representation of viruses. The profile should contain enough information to confirm the identity of the virus, but not enough information to reverse engineering the virus (i.e., obtain usable virus code from the profile representation). To be of the most use to users and researchers, the method used to represent viruses as well as a populated database of virus maps must be placed in the public domain.

The Content of the Map/Profile

In the simple case, the code profile representation describes constant sections of the virus. The section description includes the length, location, and a one-way cryptographic checksum. A description is provided for each contiguous, constant sections of code (i.e., code or constant data) in the virus. If all sections match, the executable is infected with the virus in question.

Self-garbling viruses may impose additional requirements. Encrypted viruses will require decryption before the constant sections can be checksummed. The map must include a description of the decryption technique and specify the sections to which it must be applied. Self-modifying viruses may require additional work to restore the virus to the "base case." Again, this must be performed before checksum calculation. This information must also be provided in the profile.

Precise Identification Tool Usage

A precise identification tool would have two primary uses. The first usage would be in assisting virus researchers in determining if a sample is one of a known and analyzed virus. The second usage is for users, who have detected a virus infection, to perform an accurate cleanup procedure. In either case the infected file would be compared against one or more profiles to see if the sample can be identified precisely. To make the tool most useful, no prior manipulation of the infected file should be necessary.

This implies the profile should contain information for locating the virus code inside of an infected file and information necessary to degarble self-garbling viruses.

Survey of Techniques

While there is no standard technique for the precise identification of computer viruses as of yet, there are a variety of techniques which have been used to achieve this objective. These techniques have been developed at University of Warsaw, the Bulgarian Academy of Sciences, University of Karlsruhe, and IBM's Virus Research Center. The trade-offs between these techniques are discussed, as well as an analysis of what is missing from the current techniques.

University of Warsaw

Members of the University of Warsaw, including Andrzej Kadlof, Editor-in-Chief of PCvirus Bulletin, developed the Virus Map Format. Maps of this format are distributed as part of the "Virus Identification Card" which is a regular feature of the PCvirus Bulletin. (The technique has been in use for over a year.) The virus map divides virus code into four types of fields. The four categories are:

- V - virus code
- W - working area
- C - constants
- G - garbage

Block sizes are limited to 256 bytes. If a block is larger, it is subdivided into smaller blocks of the same kind. Checksums are computed for blocks of type V, C, and G. Decimal dumps of blocks of type C or G are sometimes included.

offset	block type	length	control sums
0000 (0)	virus code	8	AE0C
0008 (8)	working area	13	
0015 (21)	virus code	373	907F 82ED
018A (394)	constants	46	E34B

```
dump:  59 6F 75 72 20 50 43 20 69 73 20 6E 6F 77 20 53 Your PC is now S
       74 6F 6E 65 64 21 07 0D 0A 0A 00 4C 45 47 41 4C toned!.....LEGAL
       49 53 45 20 4D 41 52 49 4A 55 41 4E 41 21       ISE MARIJUANA!
---------------------------------------- 0001C178:0002A4C3 --------
```

Checksums are computed with the algorithm used by the Polish Section of Virus Information Bank. The map contains records for each block. Each record contains

fields specifying offset from the beginning of the virus (in both hex and decimal), the block type, the length of the block, and a checksum value for the block when appropriate.

This tool is the precursor to some of the other tools discussed in this paper. As such, this tool may appear to be the least ambitious. However, this technique is sufficient to precisely describe the vast majority of PC viruses.

There are certain limitations, though. This tool, as it currently exists, cannot be used in an automated fashion. As an example of this, encrypted viruses must be decrypted before mapping can occur; decrypting information is not included in the Virus Map. The strength of the checksum technique is unclear. The 256-byte blocksize is an unnecessary constraint, but does not seem to pose any particular problem.

Bulgarian Academy of Sciences

Members of the Laboratory of Computer Virology in Sofia, Bulgaria developed the Virus Identification Program, or VIP. VIP was designed by Vesselin Bontchev, using the concepts in Kadlof's Virus Map. VIP was primarily intended for identifying and documenting simple viruses.

VIP has several modes; the most important are the verification mode and the map building mode. In the verification mode, VIP will compare an executable to a particular virus map to verify (or refute) that the executable is infected by that virus. In the mapping mode, VIP compares several executables which are infected by the same virus. The end result is a complete virus map.

The automated mapping is not fool-proof. It must work on a set of assumptions, and when these assumptions are incorrect the mapping process will produce an incorrect map. The assumptions are:

- that the executables are all infected by the same virus; and
- that all variable portions of the virus will be different in the different samples.

The first assumption is simply a matter of good laboratory procedures. The second assumption cannot be guaranteed, though. The assumption is probably correct if the executables supplied to VIP are different in nature (size, infection environment) and large in number. In any event, the map should be used to guide disassembly and verify the result of the automated mapping process.

The VIP map format includes some additional information in the header. This information includes the virus name, length, and a description of the relative position of the viral code in an infected executable.

Name <any_text>
Length <number>
From <base> <number>
<offset> <block_type> <length> [<chksum>]
{<offset> <block_type> <length> [<chksum>]}
[end]

<number> is [$][+|-]<digits>
<base> is Entry, Eof, Bof
<block_type> isCode, Const, Text, Var, Junk

VIP provides certain enhancements to the Kadlof Virus Map. Its greatest contribution is in the mapping process, though. As the number of new variants continues to explode, demands on the time of the research community will continue to grow. Realizing the full potential of precise identification will require maintenance of an up-to-date database of maps.

VIP addresses a narrow class of viruses (non-encrypted, non-garbled), but this class includes the majority of known variants. This is basically the same segment of viruses that are addressed by the PCvirus Virus Map (minus encrypted viruses). One enhancement under discussion for VIP is the handling of encrypted viruses. A decryption record would be added to the map. A meta-language describing decryption is envisioned; the record would include the decryption parameters and parameters. VIP may require separate maps for COM and EXE infections by the same virus.

University of Karlsruhe

Members of the Micro-BIT Virus Center are developing a very ambitious tool that has application in the precise identification field. This tool decomposes the virus code into a tree representation. At the leaves are assembly language instructions. As the decomposition progresses from the root, sections of code are determined, such as: replicator, payload, encryptor, etc. Replicator decomposes into trigger, target selector, etc. This tool requires extensive analysis to build the initial tree. This tool is also useful for determining genealogy of viruses. Heuristic analysis can be invoked to determine the similarity of sections of code.

This tool can be used to pinpoint code shared between variants of a single virus family, as well as performing precise identification with high assurance. Currently, the time required to create the tree representation is prohibitively high.

This tool is not suitable for public domain release, do to the fact that it stores the actual virus code. It has many uses in a strictly research environment, including detailed code analysis and comparison.

IBM

Members of IBM's High Integrity Computing Laboratory have developed a tool called VERV. VERV contains a high level virus description language. The tool currently supports only virus verification. Additional features for virus removal are being designed. Records specifying decryptors are included for self-garbling viruses. VERV also provides hooks for when the virus requires actions that cannot be described in the high-level language.

The file from which VERV reads virus descriptions consists of a number of virus-description blocks. Each block has the following structure:

One or more VIRUS records
A NAME record
One or more LOAD records
Zero or more DEGARBLE and related records
Zero or more ZERO records
One or more check records
Zero or more REPAIR blocks

For instance, the block for the Slow-1721 virus currently looks like this:

```
VIRUS   slow slow-1721
NAME    the Slow-1721 virus
LOAD    P-COM 0 6B4
LOAD    S-EXE 0 6B4
DEXOR1 001E 06AD 0012 0000 ; Degarble the code
DEXOR1 00EB 0159 0061 0001  ; and the data area
ZERO    0012 1              ; Zero the one-byte code-garble key
ZERO    0061 1              ; and the data-garble key
CODE    0000 00EA 38d5dc08  ; Code up to first data area
CONST   0144 014E 0ff22ad9  ; COMMAND.COM
CODE    015A 063C 74e00962  ; Code between data areas
CODE    0657 06AD ad3b0b41  ; After the second data area
```

The VIRUS records simply give a list of one-word aliases for the virus that are used on the command line to tell VERV which virus to look for. These aliases are not the full primary name of the virus (that is given on the NAME record); they are just short abbreviations that the user can use on the command line. LOAD records tell what offset within the suspect file to load and how much of the file to load. DEGARBLE records tell how to decrypt or degarble when necessary. ZERO records tell where to blank out data in the virus before checksumming occurs. CODE, CONST, and TEXT records specify sections of constant code where checksumming is performed.

VERV covers most, if not all, of the major aspects of what a precise identification tool should do. It operates on suspect files with no prior manipulation of the file. Additionally, VERV will process encrypted or garbled files. In a secure research environment, VERV can also be run in a byte-for-byte comparison mode, as opposed to a mode utilizing checksums.

Such features as REPAIR blocks are not needed for the identification task, but show the sort of functionality a commercial product based on this technology might have. The major drawback to this technique appears to be the degarbling technique. The map itself does not specify how the degarbling should proceed, but refers to hard-coded routines in the VERV program. The remainder of the map is straightforward, providing a concise description of the virus.

Future Work

These projects provide a sound basis for the next step(s) in precise identification. Map-based identification is a better choice for virus identification. Tree-based solutions hold more promise for virus analysis. In the authors' opinion, an eventual solution will require items drawn from each of the Map-based projects.

Kadlof's Map-based identification concept, as enhanced in VIP and VERV, can provide the basis for precise identification of all viruses. The VIP concept of automated map building is essential if researchers are to keep up with the flood of new viruses. The ZERO command in VERV is not strictly necessary. The result is a more complicated map, with additional blocks and checksums. However, a MASK command, zeroing variable bits in a partially constant string may prove useful.

Degarbling may be a remaining hurdle. The meta-language based degarbling technique envisioned for VIP would be a more powerful solution than hard-coded routines; however, the hard-coded routines are clearly simpler in the short run. One interesting concept in the degarbling area is to use the virus's own degarbler for the task. This is an interesting concept, but poses a variety of implementation problems and risks due to the use of the virus's own code.

Work in precise identification field should result in the following items:

- a standard, published, map format;
- a public domain set of maps for all viruses found in the wild;
- automated map building tools; and
- map verification tools.

If a hard-coded degarbling technique is used, a public domain set of degarblers may also be required.

References

Chess, David M., "Virus Verification and Removal", Virus Bulletin, November 1991.

Triffonof, Ivan and Ivaylo Hadjiatanassov, "Welcome to VIP!", Virus Identification Program documentation.

Fischer, Cristoph, "Research Activities of the Micro-BIT Virus Center: A Virus Preprocessor", Proceedings: Fifth International Computer Virus & Security Conference, 1992.

Kadlof, Andrzej, "Virus Verification and Removal", in Virus-l Vol. 4, No. 226

PRIORITIES FOR LAN SECURITY - A CASE STUDY OF A FEDERAL AGENCY'S LAN SECURITY

Shu-jen H. Chang
Room A216, Technology Building
National Institute of Standards and Technology
Gaithersburg, Maryland 20899

1 INTRODUCTION

This paper takes the reader inside a federal agency Local Area Network (the agency LAN) for a risk assessment. Based on the methodology outlined in "*A Local Area Network Security Architecture*" [1], the agency LAN is assessed for potential security threats and vulnerabilities. Security services and mechanisms are recommended for countering the vulnerabilities and threats that are found. A preliminary cost estimate for implementing these services and mechanisms is made, and a prioritized list of security mechanisms referred to as *"Priorities for LAN Security"* is proposed for the agency LAN. It is believed that the risk based approach described in this paper can be applied to other LANs to derive other LAN specific priorities for security.

1.1 Research Approach

A number of federal agencies and contractors have developed risk assessment methodologies [2] and guidelines [3], some of these have been automated [4]. However, publication of actual assessment is few, especially for assessment carried out for a specific LAN. To develop the agency specific *Priorities for LAN Security*, the following steps are taken:

Step 1 - Define the Agency LAN Configuration

In this step, the agency LAN configuration including the LAN devices, services, and other LAN resources is presented. The objectives for this step are to classify LAN resources, determine assets, and decide what the LAN will be used for and what needs protection.

Step 2 - Assess Risks for the agency LAN

This step assesses the agency LAN for potential threats and vulnerabilities found in most computer networks. A risk factor is computed for each threat and vulnerability found based on the probability of the threat occurring and the cost or loss due to this threat.

Step 3 - Select Security Services and Mechanisms for the Agency LAN

This step recommends appropriate security services and mechanisms for the specific threats found in the agency LAN. The implementation cost of each suggested mechanism is also estimated.

Step 4 - Develop Agency Specific *Priorities for LAN Security*

This step takes the threats found in step 2, the security mechanisms recommended in step 3, along with the cost estimates, and determines a cost-effective way of enhancing security in the agency LAN. The final product of this step is a prioritized list of proposed security mechanisms for the agency LAN.

These steps are described in the following sections.

2 THE AGENCY LAN CONFIGURATION & CAPABILITIES

2.1 The Agency LAN Configuration and Management

The agency LAN is a multi-server Banyan* network consisting of 17 servers and roughly 450 PCs. Each server runs a Virtual

*Mention of specific products names is for description purposes only and does not constitute a NIST endorsement.

Networking System (VINES), has one or more hard disk(s), a tape backup unit, an internal uninterruptable power supply (UPS) unit, LAN card(s), serial communication card(s), and network printer(s). The servers are spread over five agency centers, and connected via 9.6 or 19.2 kb leased lines between the centers. One of the centers hosts an HP3000 minicomputer, while another center hosts an IBM 3084/4381 mainframe for daily operations. The agency LAN operates over twisted pairs and fiber optic cables.

A division in the agency manages the LAN. The division is responsible for the physical LAN, its configuration, and providing LAN-wide services such as electronic mail, mainframe access, and modem pool services. Local system administrators take care of administrative functions such as user administration, local file services, and backup of the group's file services. The information on the layout, configuration, and network services of the agency LAN is provided by the LAN managers, much of the information is also supplemented by documents.

2.2 The Agency LAN Capabilities

In the discussion that follows the LAN services provided for agency users are described along with the aspects of each service that need protection. Though not explicitly stated, it should be noted that the identification and authentication (I&A) of a user is a prerequisite before any of the services can be rendered to the user.

LAN File Services

This service provides LAN users the capability to store their own DOS files on the LAN server disk. The files can be stored in the users' private file area or a shared LAN file area. Since the server disk is shared by many users, access control is needed to protect the privacy and integrity of the information stored on the server disk.

LAN Print Services

This service allows LAN users to print documents on a network printer physically connected to a LAN server or a dedicated network printer connected to a user's PC. Output that contains sensitive information should be directed to printers that are physically protected from unauthorized users.

LAN Application Software

This service provides LAN users the capability to access applications software stored on a LAN server. Storing frequently used software on a LAN server frees up disk spaces on users' PCs, however, these shared programs should be protected against unauthorized modification. In other words, the integrity of the software must be protected.

LAN Connections to Other Hosts

This service permits LAN users to connect to other hosts for which they are authorized users. As computers are linked together to form LANs and WANs (Wide Area Networks), it has become more and more difficult to control and monitor what system one is connected to. This can pose a threat to LAN security.

Electronic Mail

One of the most frequently used network services is the capability of exchanging electronic mail; however, electronic mail is not necessarily secure. Unless protected by encryption, electronic mail is transmitted in the clear from network node to node. In addition, while the message is routed through the network, it is quite possible for someone with system privileges to access the message and modify its contents. The privacy and/or the integrity of the messages containing sensitive information must be provided.

LAN Access Through PC Dial-in

This service allows users to access the LAN from a standalone PC that is equipped with a modem and the necessary VINES software. The service is convenient for users who may be away from their offices but need to access the LAN from a PC not physically linked to the LAN. This service can raise a security issue, since an unauthorized user with the proper software may try to access the LAN by randomly guessing a legitimate user's login id and password.

Electronic Calendaring

Electronic calendaring provides an integrated scheduling tool for a workgroup. Users can mark events on the calendar and coordinate meeting schedules with fellow workers. Proper access control is essential so that only legitimate users in the workgroup can modify the events and schedules, while others may look at the events and schedules but can not modify them.

LAN FAX Capability

This service allows LAN users to send a copy of a document stored either on the user's PC or on the server disk to any standard FAX machine. If the document contains sensitive information it should not be faxed without protection.

3 RISK ASSESSMENT OF THE AGENCY LAN

The security in a VINES network is provided by two VINES services: the StreetTalk naming service, and the VANGuard security service [5]. StreetTalk is a distributed directory service that maintains the names and attributes of all critical network resources. It enables other network services to enforce user access control and provides dynamic naming consistency over the network. VANGuard is a security service that enforces network security and authenticates all requests to access, add, or modify information.

3.1 Vulnerability/Threat Assessment of the Agency LAN

In this section, eight major threats to a computer network are reviewed for possible presence in the agency LAN. The primary vulnerabilities related to each threat, the likelihood of the threat occurring at the agency LAN, and the potential damage due to this threat are addressed. If VINES provides some protection for such a threat, the provided security features are presented. It should be noted that when a security feature is provided, the threat assessment should contain the assessment of the vendor's implementation, and the evaluation of the usage by the agency LAN users. A threat assessment is not complete without looking at both aspects; however, since the implementation contains much vendor proprietary information, this study will focus on the evaluation of the usage of the security features at the agency. The vulnerability/threat assessment is carried out mostly by the author by interviewing the agency system administrators and users and analyzing the information gathered, while the agency LAN managers are consulted from time to time. This is especially true in assessing the likelihood of threats occurring at the agency and the potential damage, which are each rated as: low (1), moderate(2), or high(3). The results of the assessment are summarized in Table 1.

3.1.1 Threat of Unauthorized LAN Access

This threat is usually caused by a weak or non-existent identification and authentication (I&A) process. The primary system vulnerabilities related to this threat are:

A. Lack of or Insufficient I&A scheme

VINES implements its I&A scheme through the StreetTalk and VANGuard services. When a user attempts to log in to the LAN, the user types in his/her StreetTalk name and password at the keyboard. The password is encrypted before it is transmitted over the LAN to the VANGuard service for authentication. The VINES password encryption algorithm is based on an one-way hash algorithm proposed by Purdy [6]. The rationale behind using this algorithm is that it is very difficult to determine the plaintext password even if the encrypted password is known. Therefore, I&A is not considered a system vulnerability.

B. Poor password choices or management

By default, VINES does not require the use of a password, however, system administrators have the option of requiring a minimum length password. At the agency, a six-character password is required. Users are allowed to change their own passwords, and when a password expires in 60 days, the user is forced to change that password. One deficiency, however, is that VINES does not check for easily guessed passwords. It is up to the users to choose adequately random passwords. The likelihood of unauthorized LAN access due to poor password choice is considered high, and the potential damage of this threat is also high.

C. Known system holes

This is not evaluated for the agency LAN since no proprietary implementation information is available.

D. Misplaced trust in other hosts

In some operating systems, certain networking software require a trusted remote host. To simplify operations for users, some system administrators may configure the server to trust all the PCs on the LAN, thus allow network access without user authentication. This is not a vulnerability in the VINES network, since all network access and service requests must be authenticated in real-time.

E. Poor physical control of LAN devices

Although the agency is in guarded buildings this does not imply that security is always tight. The servers are placed in rooms that are locked at night, but not locked at all times because users desire easy accessibility to the network printers connected to the server. Unauthorized server access can cause potentially high damage; however, all server consoles at the agency are password protected. The LAN activity is also monitored during working hours; hence the likelihood of unauthorized server access is considered "moderate".

Many user PCs at the agency are not password protected at boot time. When users are away from their PCs, they tend to leave them unlocked, even though they may still be logged in to the LAN. Therefore, the likelihood of unauthorized LAN access due to this vulnerability is high, as is the potential damage.

In order for a user to dial in to the agency LAN, the user must have access to a PC, a modem, the VINES software, and a legitimate user account and a password. For a hacker, a PC with a modem, and the VINES software are easy to obtain. The difficult part is guessing a legitimate user's id and password, provided that no default identifications such as "system", "sysadmin", "maint"...etc. are used in the LAN. The system administrators can make this guessing more difficult by setting up a user dial-in access list and restricting dial-in access to only the users who have the need. Such a list is not presently set up at the agency.

F. LAN access passwords stored in batch files on PCs

To log in to VINES, a user has to type in his/her StreetTalk name and password interactively. However, a batch file can be set up to contain the user's identification and password. When the user tries to login to the LAN, the batch file is supplied to the "login" command using the DOS standard input redirection feature. The "login" program will then receive its input from the batch file instead of the standard input device. This practice should be prohibited since it bypasses the system's only security check. The redirection of standard input and output is a DOS feature, not a VINES feature. Few users at the agency store their passwords in batch files, however, it is a vulnerability that can easily be exploited and result in unauthorized LAN access. The likelihood of this threat is rated "moderate".

G. Login attempts not recorded

Login attempts to VINES are recorded, and the system logs are examined periodically by the agency LAN managers. However, with 17 servers currently in operation, it is difficult to examine all the logs closely without the help of some automated tools. Thus the threat of unauthorized LAN access due to infrequent log scanning is rated "moderate".

3.1.2 Threat of Unauthorized Access to LAN Resources

Although one of the advantages of using a LAN is that many LAN resources can be shared among users, not all resources need to be made available to every user. Unauthorized access to LAN resources usually results from the fact that the access rights are not properly assigned, or the access control mechanism lacks granularity.

VINES controls access to network resources through two mechanisms: *adminlist*, and *access rights list* (ARL). An *adminlist* is set up for each group to identify the users who have administrative privileges in that group. Each server on the network also has a corresponding *adminlist*. With VINES, there is no "super user" who can override security throughout the entire network.

VINES uses ARLs to protect directories on the server disk, network printers, and other network resources from unauthorized access. Each ARL consists of StreetTalk names that describe who can use a resource. For printers and host connections, a user either has access to the resource or not. For directories on the server disk, the ARL not only specifies who can access information in the directory, but also the access level permitted.

In general, the responsibility of guarding against unauthorized access to LAN resources falls heavily upon the system administrators who are responsible for setting proper ARLs for LAN resources in their jurisdictions. Currently there are no ARLs set up in the agency for connecting to the IBM/IIP hosts, however, users who wish to access the hosts still need to pass the security imposed by these hosts. The likelihood of unauthorized host access is considered moderate. For file service, most system

administrators have set up proper ARLs, thus the possibility of unauthorized access to other users' files is low. File sharing between groups is possible but rare at the agency, since it requires the system administrator's permission. File sharing between users in different agency centers is not practical due to (serial) line speed considerations.

3.1.3 Threat of Compromise of LAN Data and Software

This threat usually stems from the lack of encryption capability and placing sensitive information in high traffic areas. The agency LAN is vulnerable in this area, since no encryption capability is currently provided by VINES other than password encryption. Though the agency does not process any classified information, certain information is considered sensitive and should be protected. The threat of compromising LAN data and software is considered moderate.

3.1.4 Threat of Unauthorized Modification to LAN Data and Software

Since ARLs are normally set up for the agency file services, the likelihood of unauthorized modification is considered low. However, the threat can not be ruled out completely. If the threat does occur, early detection is crucial so that unauthorized modification can not travel throughout the LAN. A cryptographic checksum mechanism can be used for this purpose [7]. Currently no such a mechanism is used for protecting the integrity of LAN data and software. The likelihood of unauthorized modification without detection is considered "moderate".

The agency has a policy regarding virus detection. All microcomputers are required to have commercial virus detection software installed and resident at all times. The threat of unauthorized modification due to virus infection on the LAN can be greatly reduced if the software that users upload to the servers is free of known viruses. The likelihood of this threat occurring at the agency LAN is considered moderate.

3.1.5 Threat of Compromise of LAN Traffic

Confidentiality of LAN traffic can be provided in two ways. One is physically securing the communications media and equipments; the other is encrypting sensitive LAN traffic. At the agency, the likelihood of compromising LAN traffic is considered moderate. Though communications hardware is generally secured, network traffic can still be intercepted with the right equipment. Furthermore, no encryption capability is currently provided for the network traffic.

3.1.6 Threat of Unauthorized Modification to LAN Traffic

Unauthorized modification to LAN traffic is less likely not only because network traffic is checksummed in compliance with the specifications of the IEEE-802.3 Standard, but also because it is difficult to modify LAN traffic in real time without being detected.

3.1.7 Threat of Spoofing of LAN Traffic

Spoofing the LAN traffic involves *(1) the ability to receive a message by masquerading as the legitimate receiving destination, or (2) masquerading as the sending machine and sending a message to a destination* [1]. To masquerade as a receiving machine, the LAN must be made to believe that the address of the machine matches the destination address on the message being sent. However, LAN traffic can be intercepted by connecting a network monitoring device to a node on the LAN.

Due to the real-time authentication required by VANGuard for any service request, it is difficult to masquerade as a message sender and attempt to retransmit a message, especially when each session is timestamped.

3.1.8 Threat of Disruption of LAN Services

Network disruption can be physical or logical. Physical disruption is related to system shutdown or hardware problems; logical disruption is caused by heavy network load or contaminated service software.

Many natural and environmental threats can disrupt a computer network, for example, fire, hurricane, tornado, flood, earthquake, and thunder storms. These threats are considered small for the agency LAN. Nevertheless, it is essential to keep the LAN cable map and other configuration information up to date, and have a comprehensive contingency plan in case an unexpected disaster does occur. At the agency, such information may not always be up to date, possibly because the likelihood of this threat to the agency LAN is low. Hardware problems and unintentional human accidents are more likely to bring down the agency LAN. Fortunately, the Ethernet hardware at the agency is monitored so that early detection of hardware problems is possible. To prevent

Table 1. Risks of the agency LAN

Potential Threat	Vulnerability	Likelihood of Threat	Potential Damage	Risk
Unauthorized LAN access	Poor password choice	High (3)	High (3)	High (9)
Unauthorized LAN access, Disruption of LAN services	Insufficient physical control on servers and easy accessibility of server key	Moderate (2)	High (3)	High (6)
Unauthorized LAN access	Lack of physical control on PCs, or unattended PCs logged in to LAN	High (3)	High (3)	High (9)
Unauthorized LAN access	User dial-in list not properly set up on servers	Low (1)	Moderate (2)	Low (2)
Unauthorized LAN access	LAN access passwords stored in batch files on unprotected PCs	Moderate (2)	High (3)	High (6)
Unauthorized LAN access	Login attempts not logged or log files not closely examined	Moderate (2)	High (3)	High (6)
Unauthorized host access	ARL not properly set up for host connections	Moderate (2)	High (3)	High (6)
Unauthorized file access	ARL not properly set up for file service	Low (1)	High (3)	Moderate (3)
Compromise of LAN data, software, & traffic	Lack of encryption capability	Moderate (2)	High (3)	High (6)
Compromise of LAN data & software	Easy accessibility to information resources	Moderate (2)	High (3)	High (6)
Unauthorized modification to files	Lack of control on uploading user software to shared area on server disks	Low (1)	High (3)	Moderate (3)
Unauthorized modification to LAN data & software	Lack of modification detection mechanism	Moderate (2)	High (3)	High (6)
Virus attack in the LAN	Virus detection software not diligently used	Moderate (2)	High (3)	High (6)
Disruption of LAN services	LAN traffic not closely monitored	Low (1)	High (3)	Moderate (3)
Difficult/slow disaster recovery	LAN cable map and configuration information not kept up to date	Low (1)	High (3)	Moderate (3)

unintentional human errors or accidents, physical access to servers and crucial communications components should be controlled. At the agency, the wiring closets are key locked and only authorized personnel have access. However, some servers are easily accessible to general users. An obvious vulnerability is the easy accessibility of the server key, which contains the server's serial number and other crucial information for server authentication and operation. Any person who can access a server can pull its server key and bring down the server instantaneously. If the key is stolen, the server can not function until a new key is replaced

by Banyan. It is recommended that a sign be posted near the server's power cable warning users not to touch or move the power cable.

Network services can be disrupted even if the LAN is physically connected. Such logical disruption can occur due to a virus on the LAN, software erasure, heavy network usage, or service being accidentally stopped. Because of the agency's policy on virus detection, hopefully a virus outbreak is less likely. Network service disruption due to the erasure of the service software is unlikely since this requires system administrator privilege. Heavy LAN usage is another factor that can degrade the LAN performance to an unacceptable level. The LAN traffic should be closely monitored so that an unusual traffic pattern can be detected early and proper action taken. Network management tools are installed in all agency servers and used for monitoring the LAN. The likelihood of disruption of LAN services is considered low due to network monitoring by LAN managers.

3.2 Risk Analysis of the Agency LAN

Table 1 summarizes the threat assessment of the agency LAN. In this section, a risk is calculated for each threat and vulnerability found in the agency LAN. The risk is determined by ***the likelihood of the threat occurring*** and ***the expected loss incurred given that the threat occurred*** as follows:

risk = likelihood of threat occurring x loss incurred

The *likelihood of the threat occurring* and *the loss incurred* are both quantified and each assigned a rating of low, moderate, or high corresponding to the number 1, 2 and 3. For example, if the likelihood of users choosing easy passwords is high (assigned value 3), and the potential damage caused by unauthorized LAN access due to this vulnerability is high (again assigned value 3), the risk is determined to be:

risk = 3 x 3 = 9.

With this assignment, a risk can have these values: 1, 2, 3, 4, 6, and 9. The risk is normalized as follows: a risk in the range of 1 to 2 is considered a low risk, in the range of 3 to 5 a moderate risk, and in the range of 6 to 9 a high risk. The risks to the agency LAN are computed and included in Table 1.

4. SECURITY SERVICES AND MECHANISMS FOR THE AGENCY LAN

Based on the risk analysis performed in section 3, security services and mechanisms that are appropriate for reducing risks from the agency LAN are recommended, and implementation costs are estimated. Table 2 shows the services and mechanisms that are recommended for the agency LAN. It is quite possible that several mechanisms can be used to counter a single threat; in that case, multiple mechanisms are listed. It should be noted that a clearly stated security policy, a well defined set of security procedures, together with adequate user training are essential for achieving and maintaining a secure LAN environment. These mechanisms can be applied to practically every threat listed in Table 2.

Also shown in the table is the cost estimate of implementing a particular mechanism in the agency LAN. Frequently, the decision on whether to use a certain mechanism is largely influenced by the cost of implementing such a mechanism. The cost is estimated based on the amount needed to purchase/develop and implement each mechanism in the agency LAN. Similar to the computation of risks, implementation cost is normalized; a 1 indicates a mechanism with a low cost, a 2 indicates a moderate cost, and a 3 indicates a high cost.

5. PRIORITIES FOR AGENCY SPECIFIC LAN SECURITY

Given the cost estimates of the security services and mechanisms, this section determines the priorities of implementing these services and mechanisms that are most cost effective for the agency LAN.

The priority of implementing a specific mechanism can be determined by examining the risks involved and the implementation cost as follows:

priority = risk/cost

The higher the ratio, the higher is the priority for implementing the particular mechanism. Deciding the implementation sequence of security mechanisms is a subjective process, especially when several mechanisms are prioritized roughly the same. Implementation cost may be a dominant consideration factor, however, there may be other concerns, requirements, or policies

that may mandate that a specific mechanism be implemented prior to others. For this reason, the implementation sequence may vary depending on the situations.

Table 2 shows the prioritized list of recommended mechanisms in decreasing order. It can be seen from the table that the threat of unauthorized LAN access due to poor password choice registers the highest priority. This is because the risk of unauthorized LAN access is high and the cost to prevent it is low. The proposed ***Priorities for LAN Security*** have been delivered to the agency for consideration for future implementation.

Table 2. Priorities for LAN Security

Threat	Vulnerability	Risk	Service	Mechanism	Cost	Priority
Unauthorized LAN access	Poor password choice	High (9)	I & A	Use software to check for easily guessed passwords	Low (1)	9
Unauthorized LAN access	Poor password choice	High (9)	I & A	Use random password generator	Low (1)	9
Unauthorized LAN access	Lack of physical control on PCs, or unattended PCs logged in to LAN	High (9)	I & A	Use PC/keyboard locking mechanism	Low (1)	9
Unauthorized LAN access	LAN access passwords stored in batch files on unprotected PCs	High (6)	I & A	Set policy & prohibit such practice	Low (1)	6
Unauthorized LAN access	Insufficient physical control on servers	High (6)	I & A	Physical control and close server monitoring	Low (1)	6
Unauthorized LAN access	Login attempts not logged or log files not closely examined	High (6)	Logging & Monitoring	Use automated software tool to scan log files	Low (1)	6
Unauthorized host access	ARL not properly set up for host connections	High (6)	Access control	Set up ARLs based on need	Low (1)	6
Compromise of LAN data & software	Easy accessibility to information resources & lack of encryption capability	High (6)	Access control	Physical control	Low (1)	6
Disruption of LAN services	Insufficient physical control on servers and easy accessibility of server key	High (6)	Monitoring	Physical control and close server monitoring	Low (1)	6
Unauthorized LAN access	Poor password choice	High (9)	I & A	Set policy & provide user training	Moderate (2)	4.5
Unauthorized LAN access	Lack of physical control on PCs, or unattended PCs logged in to LAN	High (9)	I & A	Set policy & provide user training	Moderate (2)	4.5
Unauthorized LAN access	Poor password choice	High (9)	I & A	Use token-based access control system	High (3)	3

Threat	Vulnerability	Risk	Service	Mechanism	Cost	Priority
Unauthorized LAN access	Poor password choice	High (9)	I & A	Use biometric system	High (3)	3
Unauthorized modification to data & software	Virus detection software not diligently used	High (6)	Data Integrity	Set policy & provide user training	Moderate (2)	3
Compromise of LAN traffic	LAN traffic transmitted in plaintext	High (6)	LAN message confidenti-ality	Secure LAN medium & equipments	Moderate (2)	3
Unauthorized file access	ARL not properly set up for file service	Moderate (3)	Access control	Set up ARLs based on need	Low (1)	3
Unauthorized modification to data & software	Lack of control on uploading user software to shared area on server disks	Moderate (3)	Access control	Set up ARLs properly	Low (1)	3
Disruption of LAN services	LAN traffic not closely monitored	Moderate (3)	Monitoring	Frequent monitoring	Low (1)	3
Difficult/slow disaster recovery	LAN cable map and configuration information not kept up to date	Moderate (3)	Logging & Monitoring	Keep LAN configuration information up to date	Low (1)	3
Compromise of LAN data & software	Lack of encryption capability	High (6)	Data confidenti-ality	Provide encryption capability	High (3)	2
Compromise of LAN traffic	LAN traffic not encrypted	High (6)	LAN message confidenti-ality	Encrypt sensitive LAN traffic	High (3)	2
Unauthorized LAN access	User dial-in list not properly set up on servers	Low (2)	Access control	Set up ARLs based on need	Low (1)	2
Unauthorized modification to data & software	Lack of control on uploading user software to shared area on server disks	Moderate (3)	Data Integrity	Use Message Authentication Code on sensitive data/files	High (3)	1

REFERENCES

[1] Carnahan, Lisa, *A Local Area Network Security Architecture*, Jan. 1992.

[2] *U.S. Department of Energy Risk Assessment Methodology*, reprinted as NISTIR 4325, May 1990. Available from: NTIS, Springfield, VA, 22161, telephone: (703) 487-4650.

[3] *U.S. Department of Justice Simplified Risk Analysis Guidelines*, reprinted as NISTIR 4387, Aug. 1990. Available from: NTIS, Springfield, VA, 22161, telephone: (703) 487-4650.

[4] *Proceedings of the 4th International Computer Security Risk Management*, Model Builders Workshop, Aug. 1991.

[5] Banyan Systems Incorporated, *VINES Architecture Definition*, pp. 4-3 to 4-8, Aug. 1988.

[6] Purdy, George B., *A High Security Log-in Procedure*, Communications of the ACM, Volume 17, No. 8, pp. 442-445, Aug. 1974.

[7] *Computer Data Authentication*, National Bureau of Standards, Federal Information Processing Standards Publication (FIPS 113), National Technical Information Service, Springfield, VA. 1985.

PROTECTED GROUPS: AN APPROACH TO INTEGRITY AND SECRECY IN AN OBJECT-ORIENTED DATABASE

James M. Slack
Computer and Information Sciences Department
Mankato State University
Mankato, Minnesota 56002

Elizabeth A. Unger
Department of Computing and Information Sciences
Kansas State University
Manhattan, Kansas 66506

Abstract

In this paper, we propose an approach to integrity and secrecy in object-oriented database systems based on *protected groups*. A protected group is a set of objects which will only accept messages from one or more interface objects. We show how discretionary access control, data integrity, access integrity, and the Clark-Wilson integrity model can be implemented with this approach.
Keywords: Object-oriented database, security, integrity, access control, inference control.

1 Introduction

Object-oriented databases are becoming increasingly important for many applications. Recently, there has been interest in security for these applications as well. The subject of security consists of the areas of secrecy (or confidentiality), integrity, and availability. Much of the research in object-oriented security follows the familiar lines of discretionary access control, mandatory access control, and multilevel secure database systems. Researchers are still exploring various access control mechanisms. Some possible approaches include restricting access to classes, instances, attributes, or methods. This can be done within the object-oriented database itself or with a separate authorization structure.

In this paper, we propose an approach to integrity and security in object-oriented database systems based on *protected groups*. A *one-way protected group* is a set of objects which will only accept messages from one or more interface objects. A *two-way protected group* is a refinement of a one-way protected group. We deal with one-way protected groups in this paper; see [24] for treatment of two-way protected groups.

Military security is typically concerned with control of disclosure, while commercial security is concerned with control of modification.[12, 21, 14, 15] We are interested in both kinds of security here.

The National Computer Security Center has recently defined two types of integrity in [23]:

1. *Data integrity*: concerned with preserving the meaning of information, preserving the completeness and consistency of its representations within the system, and with its correspondence to its representations external to the system. It involves successful and correct operation of both computer hardware and software with respect to data and, where applicable, the correct operations of the users in the computing system.
2. *Systems integrity*: concerned with successful and correct operation of computing resources.

Smith has identified three types of integrity:[28]

1. *Access integrity*: concerned with making sure only authorized users modify objects.
2. *Object integrity*: concerned with making sure that the value of objects are not corrupted, e.g., that an electronic funds transfer is correct.

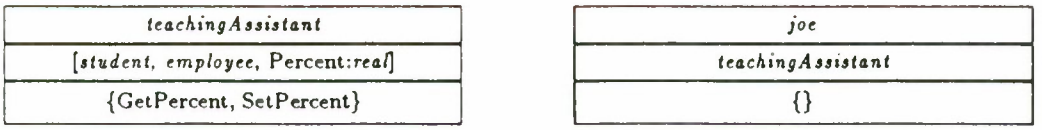


Figure 1: The object class *teachingAssistant* and the object instance *joe*.

3. *Data integrity*: concerned with making sure that the value of each object satisfies a set of validity assertions.

Each of Smith's integrity types is part of data integrity in the NCSC's definition. We are concerned with the Smith's definition of access integrity and data integrity; therefore we will use his definitions of integrity in this paper.

Several authors have suggested that the object-oriented approach may yield simpler solutions to existing security problems than the relational approach because security constraints can be incorporated within methods (e.g., [4, 21]). On the other hand, the object-oriented approach may complicate rather than simplify matters because of the increase in the number of operations and higher complexity of the structures.[19, 20, 29] In either case, it appears that the object-oriented data model will become the dominant data model in the future.[30, 17]

2 Formal Model of an Object-Oriented Database

The object-oriented data model in [25] incorporates objects, classes, methods, multiple inheritance, and encapsulation. Each class and attribute is an object in its own right, but methods are not objects. Classes are defined at the same "level" as other objects and use a simple multiple inheritance scheme. In this model, objects cannot change their structure (e.g., object instances cannot change classes). Also, the only way to access an attribute in an object is through a method of that object. Since attributes are themselves objects, they must be accessed using their own methods. The model assumes that if two methods inherited from an object's parent(s) have the same name, then they represent the same method (i.e., they are derived from the same object).

The primitive set $\mathcal{D}$ consists of the union of Z, the set of integer numerals; R, the set of real numerals (computer reals); S, the set of strings; B, the set {**true**, **false**}; the set of domain names: {**nil**, **integer**, **real**, **string**, **boolean**}; and the empty domain name *base*. We also assume the existence of $\mathcal{A}$, the countably infinite set of symbols for *attributes* and $\mathcal{I}$, the countably infinite set of *identifiers*.

An *object* s_i is a triple (i, t, m), where i is an identifier, t is the *type* of the object, and m is a set of methods. We illustrate an object as a rectangular box with three compartments (see Figure 1). The topmost compartment contains the object identifier; the middle compartment contains the type; the bottom box contains the set of methods for the object.

A *class* is an object $s_i = (i, t, m)$ where t is a set-structure, a tuple-structure, or the empty domain name *base*. Attributes of a set-structured type are *unnamed*. Attributes of a tuple-structured type may be *named* or *unnamed*. Unnamed attributes are parent class identifiers. Named attributes are instance identifiers; they are used to store values. An *instance of a class* is an object (i, t, m) where t is an object class identifier and m is the empty set. A cycle cannot exist in the identifiers in object instances or in unnamed attributes in tuple-structured and set-structured o-terms. In Figure 1, the object *teachingAssistant* is a class because [*student*, *employee*, Percent:*real*] is an o-term. The object *joe* is an instance because *teachingAssistant* is an identifier.

Methods are the operations that access and manipulate attributes within objects. The set of methods encapsulated in an object defines the behavior of that object. A method is a function which takes the state of the database, an object identifier, and a list of parameters as input, and returns a new state for the database along with a return message. A different kind of return message results when a method is invoked on a class object. A method invoked on a class object s_i results in a "set" of return messages: the return message of each instance of s_i is in this set.

The state of the database consists of the set of states of all instances in the database. The state of an tuple-structured instance consists of a sequence of object identifiers for the attributes of the tuple. The

state of a set-structured instance consists of a set of object identifiers. For illustrative purposes, we will sometimes include the state of an instance in the bottom box of an instance instead of the empty set of methods. We will also include the value of the object identifier rather than the identifier itself. For example, an instance of a class *employee* with a string and a real attribute would look like this:

emp_1
employee
['John Smith',30000.0]

A *subject* in our model is an object instance which has the ability to start a message spontaneously. A message is an instance of the class *message* where the type of the message is:

[FromObj : *object*, ToObj : *object*, ACI : *ACIinfo*, MessageName : *string*, Parameters : *parameterlist*, ...]

and the messages include SendMessage, ReceiveMessage, SetACI, and GetACI. The attribute ACI contains access control information for the originating subject. The particular access control information used depends on the security and integrity model, e.g., the user identifier for discretionary access control or Clark-Wilson integrity, security level for mandatory access control. Methods SetACI and GetACI are privileged operations; only certain objects that are registered with the system may invoke them. A spontaneous message (i.e., sent by a subject) will not contain access control information. Any other message will contain the access control information of the incoming message which started the associated method.

The class *parameterlist* defines a sequence[1] of parameters, where each parameter is an object identifier. Any object may invoke the SendMessage and ReceiveMessage methods directly, i.e., without sending a message to the *message* class. (This is necessary because otherwise, an object would need to send a message to SendMessage, but this requires sending a message to SendMessage, etc.) SendMessage and ReceiveMessage are the only methods which can access the attributes in a message.

For example, suppose subject s_{sue} wishes to find out the percentage time Joe works as a teaching assistant. Subject s_{sue} invokes SendMessage to send the message $(s_{sue}, s_{joe}, \text{null}, \text{GetPercent}, ())$. In this example, the message is from subject s_{sue} to object s_{joe}, requesting s_{joe} to invoke method GetPercent with no parameters. The value null represents the default access control.

A *return message* is a message which contains a return value (i.e., an object identifier) as a parameter. The parameter list may also contain other values which identify the message as a return message. It is up to the method language to distinguish return messages from other messages, based on the parameter list.

3 One-way Protected Groups

Our secrecy/integrity mechanism is based on the idea that objects are partitioned into *protected groups*. (For the remainder of this paper, we will use the term protected group to mean one-way protected group.) A protected group can support need-to-know and any hierarchically structured set of access rights. Each protected group has one or more *interface objects* which accept messages from any source. All other objects in a protected group are *implementation objects*. An implementation object is hidden from external view and only accepts messages from an interface object of the same group.

There are several possible implementations of protected groups; we will outline one approach which requires some enhancements to the data model. This approach assumes that the system guarantees the integrity and secrecy of messages.

Each object is augmented with the identifier of the interface object from which the object will accept messages. This information is inherited by subclasses and instances. Therefore, the inheritance mechanism is protected: an object may not have parents in another protected group.

Definition 1 A *protected object* s_i is a 4-tuple (i, t, m, g), where i is an identifier, t is an o-term or an object class identifier (defined shortly) which defines the *type* of the object, i.e., it restricts the object's domain, m is a set of methods, and g is either an object identifier from which s_i will accept messages or the reserved identifier *null*, which means s_i will accept messages from any source.

[1] A *sequence* is an ordered, heterogeneous multiset. We can define a sequence in the model as a set of tuples in which each tuple contains a value (i.e., and object identifier) and a unique integer. The integers define the order of the elements. For clarity, we will write a sequence as a list of values between parentheses, e.g., ('Smith', 12.34, 'Smith').

An implementation object $s_i = (i, t, m, g)$ is an object which only accepts messages from some other object s_g. An interface object is an object s_g from which some other object s_i will accept messages. Note that an object s_i may be both an implementation and an interface object. This feature would be useful in a "layered" protection scheme, in which one or more protected groups are enclosed in a larger protected group (e.g., see [24]).

The protected object definition could be generalized to include a set of interface object identifiers, but we will keep the approach as simple as possible for now. This protected group implementation can handle more than one class object in the same group: the interface object can manage messages for each class in the group. Name conflicts among messages of the different classes can be resolved by extra parameters in the messages. Instances of primitive classes (e.g., *integer*, *real*) can be protected by making protected versions of these classes within the protected group.

4 Secrecy/Integrity Mechanism

In this section, we explain how a combined secrecy/integrity mechanism can be constructed, based on the notion of protected groups. We first define a *chain of messages* as a sequence of messages starting at a subject and ending at some object.

We require the following additional operational characteristics:

1. The system identifies and authenticates subjects.
2. The system can hide the existence of any object (e.g., classes, instances, subjects, messages) from any other object.
3. A method can return a value that is indistinguishable from the "object not found" return value from the system.

The secrecy and integrity of a protected group of objects is based in the interface object for that group. The interface object is the only object in the group which is allowed to invoke the methods GetACI and SetACI in method *message*, i.e., it is registered with the system for this privilege.

4.1 Secrecy

Let us consider secrecy first. When the interface object receives a message from some other object, that message is of one of these forms:

1. The message contains the access control information of the originating subject, or
2. The message contains no access control information, but does contain the object identifier of the source of the message.

In the first case, the access control information is known. In the second case, the interface object can obtain the access control information based on the object identifier of the source of the message. The interface object can then set all further messages in this message chain to contain the access control information of the originating subject. If the source of the message is not a valid subject, the interface object can reject the message.

The interface object also has access to the access control information of the target object of the message. This access control information may be stored in the object as additional attributes or in a separate object within the protected group. Using the access control information of the subject and the target object, the interface object can use the following general outline for each of its methods:

```
METHOD MethodName (TargetObject, OtherParameters)
BEGIN
   IF GetACI _is null THEN
      SetACI (access control information of source object)
   ENDIF
   IF GetACI compares_favorably_with TargetObject.ACI THEN
      Invoke TargetObject.MethodName (OtherParameters)
```

```
    ELSE
        RETURN ('Object not found')
    ENDIF
END
```

In this approach, secrecy is a precondition that must be satisfied before access is allowed. The implementation of the comparison operator *compares_favorably_with* depends on the secrecy mechanism and the type of access control information.

4.1.1 Discretionary Access Control

The implementation of traditional discretionary access control is straightforward in this setting. Let us consider an example based on access control lists where access is determined by the subject (user) identifier. The class *interface* is defined as follows:

interface
base
{MakeClass} $\bigcup X$
null

where X is the set of methods from all other classes in this protection group. Class *interface* works together with class *auth* to restrict access to other objects in the protection group. Class *auth* is responsible for checking whether a subject s_i is authorized to invoke a method m. It is defined as follows:

auth
[Object:*objId*,Meth:*string*,Users:*userSet*]
{OkToUse,Grant,Revoke}
interface

Both *auth* and *interface* are permitted to access and change the access control information in messages. Method MakeClass in *interface* creates new classes in this protection group. For each new class, MakeClass includes an Init method which creates instances of that class. This approach ensures that each new object in the protection group is never in an unprotected state.

Let us now add a class of subjects and a class of employees to the protection group. We use MakeClass to create the following two classes:

subject
[ID:*integer*,Password:*string*]
{Add,Delete}
interface

employee
[ID:*integer*,Salary:*real*]
{GetID,SetID,GetSalary,SetSalary}
interface

Some secure database systems have both grant and give-grant[2] capability, e.g., SeaView [16]. In these systems, user A can grant access to a method to another user if A has *grant* authority for the method. If A has give-grant authority for a method, then A can authorize another user to grant access to the method.

The grant/give-grant approach can be implemented by incorporating more information into *auth* instances:

[2] Grant and give-grant are called *control* and *control with passing ability*, respectively in [22].

auth
[Object:*objId*,Meth:*string*,Users:*userSet*, Grantors:*userSet*,GiveGrantors:*userSet*]
{OkToUse,Grant,GiveGrant,Revoke}
interface

This approach allows grant and give-grant privileges to apply to each method of a class. Finer grained control is also available by having grant and give-grant privileges associated with methods of instances. Other approaches are also possible, e.g., using rules in the Grant and GiveGrant methods of the *interface* class.

There are two common types of authorization revocations: simple and cascaded.[16, 7] A *simple revocation* of access for subject s_a removes authorization for subject s_a. A *cascaded revocation* of access for subject s_a removes authorization for subject s_a and every subject s_b, to which s_a granted the same access, and every subject s_{b_j} to which s_{b_i} granted the same access, and so forth. A cascaded revocation of a particular access permission of subject s_a can get complicated if subject s_b was granted that permission by s_a and another subject. There are several possible policies which could determine what to do in such a situation. We will assume that the appropriate policy is encoded in the CascadingRevoke method.

One way to design class *auth* to support both simple and cascading revocations is as follows:

auth
[Object:*objId*,Meth:*string*,Users:*userSet*, Grantors:*GuserSet*,GiveGrantors:*GuserSet*]
{OkToUse,Grant,GiveGrant, SimpleRevoke,CascadingRevoke}
interface

The Grantors attribute stores the set of subjects which are allowed to grant access to this method to other subjects. The GiveGrantors attribute stores the set of subjects which are allowed to pass the granting authority for this method to other subjects. Each of these attributes also stores, for each subject s_b with grant or give-grant authority, respectively, the subject s_a which gave s_b that authority. (The domain of class *GuserSet* is $\mathcal{P}(subject \times subject)$, where $\mathcal{P}(a)$ denotes the power set of a.)

The access control information of the access control information of the target object is a set of valid subject identifiers; the comparison operator is set membership.

4.2 Integrity

In an object-oriented database model, access integrity is simply a special case of secrecy where the controlled method is state-changing. With access integrity, the subject's access control information must compare favorably with the target object's access control information before the method is invoked.

An object-oriented data model based on protected groups can enforce data integrity by using rules in methods of an interface object. In [27], we propose an integrity model for object-oriented databases. Each integrity constraint is a first order predicate logic expression. The model constrains the results of functional methods. All comparison and other operators in constraint expressions are methods from the object set. The following constraint restricts the result of the method GetAge of each instance of *employee* to an integer greater than 0 and less than 120:

$$\forall e(employee(e) \rightarrow \forall a(\mathrm{GetAge}(e,a) \rightarrow (\mathrm{Less}(e,0,a) \wedge \mathrm{Less}(e,a,120)))) \quad (1)$$

The constraint model is based on first-order predicate calculus, which unfortunately makes constraints rather unwieldy. Figure 2 gives a few examples of notational conveniences from [27] which make constraints far more palatable. Using these, constraint 1 can be rewritten as follows:

$$\forall e \in employee(e.\mathrm{GetAge} > 0 \wedge e.\mathrm{GetAge} < 120) \quad (2)$$

1. $e_1 != e_2$ is equivalent to $\neg == (e_1, e_2)$.

2. $\exists x \in employee(x.\text{GetSalary} > 100000)$ is equivalent to $\exists x(employee(x) \wedge \exists a(\text{GetSalary}(x, a) \wedge \text{Less}(integer, 100000, a)))$.

3. $\forall x, y \in employee(x.\text{GetID} == y.\text{GetID} \rightarrow x == y)$ is equivalent to $\forall x \forall y(employee(x) \wedge employee(y) \rightarrow \forall a(\text{GetID}(x, a) \rightarrow \forall b(\text{GetID}(y, b) \rightarrow == (a, b) \rightarrow == (x, y))))$.

4. $\forall x \in employee_{GetSalary<10000}(e.\text{GetTax} = 0)$ is equivalent to $\forall x(employee(x) \rightarrow \forall a(\text{GetSalary}(x, a) \wedge \text{Less}(integer, a, 10000) \rightarrow \forall b(\text{GetTax}(x, b) \rightarrow \text{Equal}(integer, b, 0))))$.

Figure 2: Examples of notational abbreviations in integrity constraint expressions.

In constraints 1 and 2, GetAge is a predicate which is true if the method e.GetAge a returns a, where e.GetAge is the GetAge method accessible to object e. The predicate Less is also associated with a method in e; $\text{Less}(e, 0, a)$ is true if $e.\text{Less}(0, a) = \text{true}$. This is the most general way to write a constraint: all operations must use methods of the objects.

The primitive symbols in the language of constraint expressions include variables, constants, functions, predicates, logical connectives ($\wedge, \vee, \rightarrow, \neg$), quantifiers ($\forall, \exists$), and parenthesis. A *term* is either a constant or $f(a_1, a_2, \ldots, a_n)$ where f is an n-ary function and $a_1, a_2, \ldots, a_n$ are terms. A *formula* is $P(t_1, t_2, \ldots, t_n)$ where P is an n-ary predicate symbol and $t_1, t_2, \ldots, t_n$ are terms. If A and B are formulas, then $A \wedge B, A \vee B, A \rightarrow B, \neg A, \forall x A, \exists x A$ are formulas. A *closed formula* is one for which each variable is quantified.

Integrity constraint expressions are restricted to be closed formulas as well as *safe formulas*. Intuitively, a safe formula is one in which the variables range over the extension of the database. Constraints may only use functional methods, and may ignore the possibility that a method may fail. A failing method in a constraint is a run-time error.

Integrity constraint expressions can enforce common state integrity constraints, including domain, key, and structural constraints.[26] It can enforce value-based functional dependencies from the relational model, both within an object and across object classes.

The discretionary access control technique based on protected groups presented in section 4.1.1 can be used to enforce Clark-Wilson [6] style integrity. Methods in the interface object are the TPs and objects in the protected group are the CDIs. Access triples are stored in class *auth*. Another approach to enforcement of Clark-Wilson style integrity is the Generalized Framework for Access Control;[3, 1, 2, 13] this can be applied to the object-oriented data model based on protected groups.

4.2.1 Inference Control

Kaushik has developed a simple technique for foiling tracker attacks on statistical databases[3].[11] Vemulapalli has expanded the approach to provide a wider range of actions but uses the same underlying principles.[32, 31] For each query, the database system finds the *query set* before computing the aggregate value. The system then duplicates a tuple in the query set, deletes a tuple in the query set, or does nothing to the query set. A requirement of this approach is that if two queries q_1 and q_2 result in the same query set, the same perturbation action must be taken for both queries.

Kaushik's approach can be adapted to the object-oriented data model based on protected groups. Let Aggregate be a method in interface object *interface* which returns an aggregate value from the protected group, and suppose subject s_a invokes Aggregate. An outline of this method is as follows:

1. Determine if subject s_a is authorized to access Aggregate (e.g., by sending a message to *auth*). If not, then stop.

2. Find the query set.

3. Use a function of the query set (e.g., the size of the query set) to take one of Kaushik's three actions to perturb the data:

[3] Kaushik developed his approach in the relational model; however, the results carry over to the object-oriented database model.

- Duplicate an object in the query set. Make this dummy object visible only to s_a.
- Delete an object in the query set. This can be accomplished by temporarily removing s_a's authority to access the object.
- Do nothing to the query set.

4. Compute the aggregate value from the query set.

5. Reverse any action taken in step 3.

5 Related Work

This work is similar to work by Faatz and Spooner [8] and Jajodia and Kogan.[10] In [8], Faatz and Spooner suggest an approach to discretionary access control for object-oriented engineering databases. Each component of a project has a set of implementation objects and an interface object. To activate a method in an implementation object, a user must send a message to the interface object which then forwards the message to the implementation object.

This work is simpler than [8] in that the entire database can be enclosed in one protected group. Discretionary access, data integrity, and access integrity can be controlled from a single interface object.

Jajodia and Kogan propose a mandatory access control mechanism for object-oriented databases which is based on *message filtering*.[10] Each object is assigned a classification, and each subject a clearance as with the traditional Bell-LaPadula model. Messages are not allowed to flow from object to object directly; instead, they must pass through a message filter. The message filter lets the message go through only if the information would not flow from a high level to a low level object or subject. Our approach is more flexible because message filtering is explicitly part of the object-oriented data model rather than a built-in system function.

McDermid and Hocking have devised an access control mechanism for software development environments in [21]. In their model, the set of access modes for an object consists of the ability to invoke each method of that object. The database designer can choose the relative precedence of mandatory access control and discretionary access control. The model allows more than one policy for the same database (e.g., for guests and regular users), and one of the policies can override if there are conflicts. Apparently this is dependent on the application. McDermid and Hocking claim that this system can be generalized to an access control matrix between subjects and objects, which unifies mandatory and discretionary policies. Separation of duty rules can be enforced by specifying a policy which requires a set of users to access a method.

6 Conclusion

We have described an approach to integrity and secrecy in object-oriented database systems based on protected groups. A one-way protected group is a set of objects which will only accept messages from one or more interface objects.

We have shown how discretionary access control, data integrity, access integrity, Kaushik's inference control technique, and the Clark-Wilson integrity model can be implemented with this approach. The object-oriented data model applies to distributed object-oriented systems if the integrity and security of messages can be guaranteed between nodes.

Future work includes handling the inference problem, the aggregation problem and designing a mechanism for mandatory access control. The *aggregation problem* occurs when an individual can surmise a sensitive data value by combining several nonsensitive data values from the database.[9, 5, 18]

There are several possible approaches to mandatory access control. One approach is to follow the Generalized Framework for Access Control [3, 1, 2, 13], using a single protected group and rules to determine whether access is allowed. This approach is similar to the discretionary access control mechanism described earlier. We present a different approach in [24] using two-way protected groups. A *two-way protected group* is a one-way protected group in which each object may only communicate with an interface object of the same group, and each interface object may only communicate with other objects in the same group. An interface object of one group may be an implementation object in another group. A database is composed of several protected groups, where each protected group consists of objects from a single security level.

Acknowledgments

The work of both authors was partially supported by Department of Defense grant #5-30296. The authors wish to thank John Campbell and Howard Stainer for their support and encouragement. This work is our own, however, and does not necessarily reflect the views of these other people.

References

[1] Marshall D. Abrams, Kenneth W. Eggers, Leonard J. LaPadula, and Ingred M. Olson. A generalized framework for access control: An informal description. In *13th National Computer Security Conference*, October 1990.

[2] Marshall D. Abrams, Jody Heaney, Osborne King, Leonard J. LaPadula, Manette Lazear, and Ingred M. Olson. Generalized framework for access control: Towards prototyping the ORGCON policy. In *14th National Computer Security Conference*, October 1991.

[3] M.D. Abrams, A.B. Jeng, and I.M. Olson. Unified access control: An informal description. Technical Report MTR-89W00230, MITRE Corporation, September 1989.

[4] Philip Bernstein and et al. The Laguna Beach report. *SIGMOD Record*, 18(1):17–26, March 1989.

[5] John R. Campbell. From tuples to trusted subjects to TDI: A brief tutorial on trusted database management systems. In *14th National Computer Security Conference*, 1991.

[6] D.D. Clark and D.R. Wilson. A comparison of commercial and military/computer security policies. In *IEEE Proceedings of 1987 Symposium on Security and Privacy*, April 1987.

[7] Dorothy Denning. *Cryptography and Data Security*. Addison-Wesley, 1982.

[8] Donald B. Faatz and David L. Spooner. Discretionary access control in object-oriented engineering database systems. In *Database Security IV: Status and Prospects*, pages 73–84, 1991.

[9] D.K. Hsiao. Database security course module. In Carl E. Landwehr, editor, *Database Security: Status and Prospects*, pages 269–302. North-Holland, 1988.

[10] Sushil Jajodia and Boris Kogan. Integrating and object-oriented data model with multilevel security. In *Proceedings of IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 1990. IEEE.

[11] Nanda Kaushik. A new deterrent to compromise of confidential information from statistical databases. Master's thesis, Kansas State University, 1988.

[12] Carl E. Landwehr. Database security: Where are we? In Carl E. Landwehr, editor, *Database Security: Status and Prospects*. North-Holland, 1988.

[13] Leonard J. LaPadula. Formal modeling in a generalized framework for access control. In *The Computer Security Foundations Workshop, III*, June 1990.

[14] T.M.P. Lee. Using mandatory integrity to enforce commercial security. In *IEEE Proceedings of 1988 Symposium on Security and Privacy*, pages 114–146. IEEE, 1988.

[15] S.B. Lipner. Non-discretionary controls for commercial applications. In *IEEE Proceedings of 1982 Symposium on Security and Privacy*, pages 2–10. IEEE, 1982.

[16] Teresa Lunt, Dorothy Denning, Roger Schell, and William Shockley. The SeaView security model. *IEEE Transactions on Software Engineering*, 16(6), June 1990.

[17] Teresa Lunt and Jonathon Millen. Secure knowledge-based systems. Technical Report SRI-CSL-90-04, SRI International, Menlo Park, CA, August 1989.

[18] Teresa F. Lunt. Aggregation and inference: Facts and fallacies. In *IEEE Symposium on Research in Security and Privacy*, May 1989.

[19] Teresa F. Lunt. Discretionary security for object-oriented database systems. Technical Report SRI Project 7543, SRI International, September 1990.

[20] Frank A. Manola. A personal view of DBMS security. In Carl E. Landwehr, editor, *Database Security: Status and Prospects*. North-Holland, 1988.

[21] John A. McDermid and Ernest S. Hocking. Security policies for integrated project support environments. In D.L. Spooner and C.E. Landwehr, editors, *Database Security, III: Status and Prospects*, pages 41–74. North-Holland, 1990.

[22] National Computer Security Center. *A Guide to Understanding Discretionary Access Control in Trusted Systems*. Washington, D.C., September 1987. NCSC-TG-003 Version-1.

[23] National Computer Security Center. *Integrity in Automated Information Systems*. Washington, D.C., September 1991. C Technical Report 79-91.

[24] James M. Slack and Elizabeth A. Unger. Mandatory access control in an object-oriented database using protected groups. Submitted for publication.

[25] James M. Slack and Elizabeth A. Unger. A formal model of object structure and inheritance for object-oriented database systems. In *Proceedings of Great Lakes Computer Science Conference*, Kalamazoo, Michigan, October 1991.

[26] James M. Slack and Elizabeth A. Unger. A model of integrity for object-oriented database systems. Technical Report TR-CS-91-13, Kansas State University Department of Computer and Information Sciences, 1991.

[27] James M. Slack and Elizabeth A. Unger. A model of integrity for object-oriented database systems. In *1992 Symposium on Applied Computing*, Kansas City, March 1992.

[28] Gary W. Smith. *The Modeling and Representation of Security Semantics for Database Applications*. PhD thesis, George Mason University, Spring 1990.

[29] David L. Spooner. The impact of inheritance on security in object-oriented database systems. In Carl E. Landwehr, editor, *Database Security: Status and Prospects, II*, pages 141–150. North-Holland, 1989.

[30] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems*. Computer Science Press, 1988.

[31] Kasinath C. Vemulapalli and Elizabeth A. Unger. Investigation of output perturbation techniques. Technical Report TR-CS-91-10, Kansas State University Department of Computing and Information Sciences, 1991.

[32] Kasinath C. Vemulapalli and Elizabeth A. Unger. Output perturbation techniques for the security of statistical databases. In *14th National Computer Security Conference*, October 1991.

PROVABLY WEAK CRYPTOGRAPHIC SYSTEMS

John Higgins
Brigham Young University
Computer Science Dept., 3374 TMCB
Provo, UT 84602

and

Cameron Mashayekhi
WordPerfect Corporation, MS-M210
1555 N. Technology Way
Orem, UT 84057

Abstract

The quest for absolute verification of uncompromisable software security has led to the development of a variety of crypto-systems. Publicized over-simplification of cryptanalysis with disregard to impracticality of most of the theoretical methods proposed, has caused a wide-spread paranoia among software users. Consequently, the quest for secure crypto-systems may in some respects have been too successful. There is evidence that most of the current standard crypto-systems have complexity beyond the nominal capacity of even the most well equipped and skillful cryptanalysis including agencies of government. The wide dissemination of computationally secure crypto-systems is of great concern to some agencies of government. Export restrictions on the sale of software with cryptographic features, is a serious impediment to U.S. based software companies. What is needed is the development of crypto-systems that provide adequate security for the purpose of the user and yet not so sophisticated as to lead to government prohibition of worldwide dissemination. This paper suggests methods to modify a current system so as to meet this need.

Introduction

Research into modern crypto-systems has been directed towards obtaining systems that have mathematically demonstrable levels of total security[9][19]. These systems have proven to be computationally secure in relation to the idealized methods of massive computational cryptanalysis. However, continuous publication of theoretical papers on computational cryptanalysis of these crypto-systems has driven the software users to constantly demand higher levels of security[11]. This observation presents a serious problem for commercial developers of software. There is currently widespread consumer demand that even the most innocuous of commercial software systems, should include some cryptographic features[24]. In addition, the systems should have levels of security that are predictable and certifiable[4].

Five years ago the average user of a personal computer (PC), where the usual application is for word processing or spread sheet functions, would have been shocked at the notion that her efforts demanded a cryptographic system in order to maintain privacy. This is not surprising. There is a very natural human tendency to ascribe to any new system the effective attributes of the old. The early methodology of automobile traffic control was based on experience with

ships at sea and wagon teams. Both of these familiar models quickly proved dangerously inadequate. In the same fashion, the average PC user equated electronic files with folders within a locked filing cabinet. A prospectus generated by a spread sheet and stored on a PC was the same as one done with pencil and slide rule and filed in a locked attache case. But the analogy, relative to privacy and security, were dangerously inexact. Most PC users are now aware of this and it is this fact that impels the attempted marriage of innocuous and widely used popular software and very sophisticated cryptographic systems[24].

The distinction relative to security between electronic file and filing cabinet is now widely appreciated in the business community. The fact that obtaining access to material stored in a computer system can be much easier than obtaining the same access to hard documents stored in a locked physical system is common knowledge. Indeed, the level of publicity relative to data security, or more precisely the lack of data security in large institutions has generated an attitude of concern on the part of software users. This attitude in turn has caused expressed demands on the developers of software to add security features to many popular and widely used software devices.[21]

It is pointless to argue that much of this concern is not well founded. Data security experts know that most data security problems can be solved by the application of a relatively few and rather simple rules involving the physical surroundings of the systems and proper training of system users[21]. This fact notwithstanding, it would be economic suicide for any software company to suggest such an approach as an excuse for not including security features in their products. This is the reason why this is so rooted in the nature of competitive, consumer directed markets. In a consumer economy, businesses survive by meeting the needs of the customer. Customer needs are based in large measure not on actual physical demands but on states of mind. All knowledgeable commentators admit that an important aspect of American business genius was the recognition that vague feelings could be translated into a concrete need and this need into sales. Current software consumers have been convinced that they need the protection of add on cryptographic systems. Given this fact, no software manufacturer will prosper by trying to convince buyers that they do not need such feature when the competitors are willing to provide them[13].

Development of Cryptographic Systems

Public demand for security features as outlined in the Introduction presents the software developer with a very difficult problem. Cryptographic systems are very difficult to successfully develop. This fact is well documented[2][4][18][19][20][22]. A developer generating a popular word processor has many issues which are absolutely essential to the success of the software. It is on these essential issues that the bulk of the development effort must concentrate. In addition to these essential tools, some cryptographic features must be added since that is what the current market demands. How should the developer proceed? What system should be included? Clearly the easiest approach would be to buy an off the shelf system and incorporate that system into the software. However, all current off the shelf systems are under export control. The manufacturer can not afford to forgo the export market. Similarly the manufacturer cannot afford to generate his own cryptographic system and cannot spend the time required to

have his own modifications of current systems certified by agencies of government. These facts present the software company with a serious dilemma.

The problem of the export market probably needs some clarification. Since the vast bulk of the U.S. software industry sales are domestic it would seem reasonable for the manufacturer to use systems such as DES which are acceptable for domestic software and abandon the export market. Unfortunately the solution is not that simple. The problem of marginal return is crucial in the software industry. For software, the vast majority of the cost of manufacturing is in the development of the product. The production of the physical object conveying the software and its manual(s) are relatively insignificant. The marginal return on any sales above those that meet development costs are huge. In order to succeed, it is imperative to obtain those returns that entrepreneur risk funds in the development of software. It is not now possible for a responsible software company in today's market to limit its product distribution to the domestic market. The reason for this is that the relative cost of software development is now very large. Modern popular software demands very complex and capable programs that require large expenditure in their development. The offshore market is becoming more important in providing the vital marginal return that means the difference between success or failure. This reliance on out of country sales will become more pronounced in the future. This means that there is an urgent need for the development of cryptographic systems sufficiently strong so as to meet the needs of consumers and sufficiently weak so as to meet the certification standards for export[5].

On the issue of development it must be noted that for the manufacturer, time is absolutely of the essence. To a government agency, a six month process of certification may seem like blinding speed. For the software developer, a six week delay in the introduction of a given product may spell the difference between riches and bankruptcy. The dissonance on this issue should not be trivialized. It is very unlikely that the governmental agencies will be able to alter their behavior in order to accommodate the software companies. The realities of the current domestic market and the rise of foreign software developers suggest that competition in software manufacturing will only get worse. There is a real need for the development of standardized cryptographic systems which meet three criteria:

(i) These systems must be available to the software manufacturer at reasonable cost,
(ii) They must have the same relative level of accepted computational security as current systems such as DES,
(iii) The specific application must be capable of an expedited approval by government agencies.

The first attribute is absolutely essential. The cost of development of new systems is an unnecessary expense not really germane to the general purpose of most software. The second property will be a consumer demand. Anyone who needs a security system should know what level of security the system provides. The third requirement is the sole reason for their being a problem in the first place.

For the purposes of this paper the ban on export of capable cryptographic systems is an axiom. The argument that government ought to change its view of this matter will need to be addressed elsewhere. For the software companies the issue of which cryptographic system to use is an

immediate one. The alteration of fixed government policy is based on historic experience, a matter of years if not decades. Software companies must survive in the short run or there is no long run. For them the problem is now and the attitude of the government must be taken as in some sense fixed. While some negotiation on the margins of this matter may be possible, it seems very unlikely that there will be any near term reversal of the export ban on high level cryptographic systems.

Uses of DES

Let us first examine the issue of why an established cryptographic system is important. As noted above, software development companies are not in the cryptography business. It is an area of development strewn with hazards that most commercial ventures are not well equipped to treat [3]. The most daunting problem is that of creating a credible system. The literature of modern cryptography is full of plausible and initially promising systems that were not capable of extensive development. As the survey by Brickell and Odlyzko rather poignantly demonstrates, knowledgeable investigators have invested large amounts of time and creative energy in the invention of systems that proved to be totally unreliable[22]. It is rather shocking how frequently these capable and well intended intellectual efforts have failed. This fact is not lost on the software developer. Developing a proprietary cryptographic system is almost sure to result in disaster. The paper by Kochanski is most instructive in this regard [17]. In this effort, the author was able to do a successful cryptanalysis of five cryptographic systems appended to popular software. The analysis demanded nothing more in the way of hardware than a PC. The type of failure outlined in this article is unacceptable for a well run commercial enterprise since it casts doubt on the capability and accuracy of the entire software package. The manufacturer is not going to waste time and resources trying to develop a new cryptographic system when the experience of the last decade has proven this to be a virtually impossible task.

A related problem is that of customer acceptance. Knowledgeable customers are at least vaguely aware of the hazards inherent in the development of cryptographic systems. Even if a given company were able to develop a system whose level of security was predictable, convincing the potential customer of this fact would be difficult if not impossible. The area of cryptographic validity is another of those aspects of modern civilization where it is virtually impossible for the consumer of a product to usefully investigate the claims of the product. The fact that a given crypto-system may be difficult for the customer to break means nothing. Some external certification of quality is essential. It is for this reason that a reasonable modification of an existing accepted system such as DES is absolutely essential for the manufacturers.

This leads to the examination of some provably weakened form of DES as the basis for a relatively low level cryptographic system appropriate for incorporation in widely distributed software. The reasons for selecting DES are as follows.

1. DES has a very high level of experimental security.[3] The standard algorithm has been publicly available for some time. It has been extensively analyzed and no elementary method of cryptanalysis has been discovered[3]. Indeed, with the exception of a claimed chosen plain text statistical anomaly recently announced in the media[25], the only valid general method of cryptanalysis known to the

authors remains the brute force key matching method first suggested by Diffie and Hellman[14][10]. Even for somewhat smaller key spaces than the standard 56 bits, brute force cryptanalysis of DES should be well beyond the capability of all but the wealthiest and most determined of private individuals.

2. There are software implementations of DES that are relatively fast[8]. Well designed implementations when appended to most popular products should be almost invisible to the user. This is very important since while consumers want to be able to use encryption in a variety of ways with existing software, they are not willing to pay much if anything in software performance cost to obtain this feature.

3. DES is cheap. The algorithm is public[23]. There are many off the shelf and public domain implementations of DES which software manufacturers of all sizes could quickly modify and install, if an appropriate weakened format could be designed.

4. DES should be relatively easy to weaken. By reducing the key size and perhaps the number of iterations of the fundamental transformation seems plausible that precisely quantifiable reductions in security relative to the assumed level of security of the original algorithm can be obtained[6].

5. DES ought to be relatively non threatening. Since the algorithm has been around for some time and was originally proposed by the federal government it seems plausible that agencies of government have learned to accommodate themselves to its existence. If governmental concern on the amount of time needed to break full DES is an issue, as it seems to be, then a weaker form of DES should be more acceptable than any new proprietary system that is not trivial.

The last point demands further discussion. When DES was first proposed as a standard for encryption it was widely assumed that it was inherently compromised so as to allow security agency cryptanalysis. This may or may not be the case. What is the case is that governmental agencies involved in national security seem reluctant to certify applications of DES for export. Devices such as home satellite television decoders are still export restricted when they include any of the systems that are based on the DES standard. This type of ban may just be an example of bureaucratic overkill relative to a system whose cryptanalysis is elementary given the right equipment and the right algorithms. But another interpretation of these facts does need to be addressed. As noted previously, the academic cryptanalysis community has been beating on DES for several years now. Not all members of this community can be plausibly described as being co-opted by agencies of the government of the United States. This certainly has at least the appearance of a good faith effort to discover weaknesses in the algorithm and none have been found[3]. It is arguable that the situation is just as it seems. That is, the DES algorithm is quite robust and there is no elementary trap door. If this were the case then it is not unreasonable to expect some degree of concern on the part of security agencies relative to widespread foreign sales of software using DES.

If it is in fact the case that the standard implementation of DES is a relatively secure system of encryption then for-export applications of DES will demand a provably weakened form in order to obtain export approval. Which leads to the question of demand for a demonstrably weak cryptographic system. In this regard much of the public research in cryptography is of little practical significance and leads to clearly erroneous conclusions. Someone who buys a combination lock for a tool shed does not expect to get a time-delayed bank vault door. That would be silly. The quality of the lock should be in some sense appropriate to the value of the material being protected and the construction of the general containment structure. No one can reasonably expect the encryption features of a popular software package to be resistant to all methods of cryptanalysis. The user can reasonably expect that the security features will protect against common threats. It does not take ultimate cryptographic sophistication to protect against the types of intrusions normal in security breaches. Indeed, cryptanalysis is much more difficult than it seems. Dedicated amateurs with access to reasonably powerful computer systems wrestled unsuccessfully with the Beal ciphers[14]. Less well educated amateurs with no greater computational capacity and far less time will not quickly and easily decrypt even weak forms of DES.

Short Key DES

A discussion of a modified or weakened DES requires a benchmark of analysis to which the relative weakening can be compared. This paper uses the exhaustive search methodology first outlined by Diffie and Hellman in 1977[10]. It does say something for the sturdiness of the DES algorithm that the intervening years have produced no methods significantly superior to the key search method proposed fifteen years ago. The following analysis is conducted relative to the following assumptions:

(i) the exhaustive search must be executed on available hardware of a type that is likely to be in the possession of adversaries. The ability to spend thousands of dollars on dedicated hardware is clearly beyond the capacity of all but a vanishingly small set of security adversaries[26]. A record of computer criminals exists[1]. There are virtually no entries for large criminal organizations or other such well funded and highly organized entities.

(ii) A reasonable number of known plain text characters would be available to the cryptanalysis. However, there is no capacity for a chosen plain text attack. In most low level applications, chosen plain text attack is well beyond the capacity of the typical adversary.

(iii) A single search cycle can be completed in one micro-second. While it does seem plausible that widely available hardware can be programmed to perform one algorithm iteration and comparison in time that is of the order of one micro-second, order of magnitude improvements on this standard do not seem to be easily obtained[10][17]. Something like 100 mips are currently available in standard hardware. Even if 1000 mips were to be achieved in a decade, there should be no appreciable degradation in the one microsecond standard since a

programmed DES iteration and comparison in 100 machine instructions seems difficult to achieve.

(iv) There is an elementary DES key reduction transformation that does not reduce the relative difficulty of DES cryptanalysis. This is the most tenuous assumption of the discussion. It is not at all clear that there is a method for extending a randomly selected k-bit key($k<56$) to a 56 bit key so that cryptanalysis of DES still requires exhaustive key search over the 2^k element key space. A discussion of this question is beyond the scope of this paper.

Garon and Outerbridge[12] have produced a complete an exhaustive analysis of the statistics and cost of cryptanalysis of DES by hardware and software. Again, in their case the assumption is made that the adversary has the limitless capabilties to support the cost and effort in this matter. The following table gives the time to obtain one solution, relative to the restricted key space.

Speed of Processor	Time required to exhaust the reduced keyspace of DES(40/56)			
Key-test per second	Years	Months	Weeks	Days
1 Million	1,640	19,680	85,280	596,960
2 Million	820	9,480	42,640	298,480
4 Million	410	4,740	21,320	149,240
32 Million (by 1995)	51	2,370	10,660	74,620
256 Million (by 2000)	6	1,185	5,330	37,310

Given that the problem of reduced key extension could be safely resolved, the software manufacturer and the related government agencies may be able to agree upon a standard which is mutually acceptable to all parties. If the manufacturer consented to accept a standard that was pre-defined as being acceptable for export the relevant export license could be granted in a matter of days as opposed to months. The manufacturer would then have the obligation of informing purchasers of the relative strength or weakness if you prefer, of the protection system included. The disclaimer would indicate that the purchaser was buying a product that would give adequate protection against the type of threat most commonly found in security problems. It would by implication if not directly indicate that this security feature is not proof against intrusion by the technology available to the security agencies of nation states nor perhaps to the capabilities of massively wealthy and highly organized criminal conspiracies. In any event all parties to the compact would be as well informed as current public information allows. Given current legal realities, that is about as much as can reasonably be expected[16].

Conclusion

The problem of export restrictions on popular software is a real one. This problem will become worse as the demand for encryption protection increases and the export market becomes relatively more important to the manufacturers. The mutterings of academics not withstanding,

a software company must live in the world that exists. In this world, the export of software that includes the standard implementation of DES is not permitted by government fiat. The resolution of edicts such as this may well lie far in the future. What does seem clear is that for the foreseeable future there will continue to be governmental restrictions on the export of any cryptographic system that is too capable. In this environment, a suitably weak form of a familiar and well tested system for full document encryption such as DES does seem like a plausible short term solution.

References

[1] Anne W. Branscom, "Rogue computer programs and computer rogues: Tailoring the punishment to fit the crime," *Rutgers Computer and Technology Law Journal*, vol. 16, no. 1, 1990.

[2] G. Brassard, "A note on the complexity of cryptography," *IEEE Trans. Informat. Theory*, vol. IT-25, pp. 232-233, 1979.

[3] Ernest F. Brickell, Andrew M. Odlyzko, "Cryptanalysis: A survey of recent results," in Proceedings of the IEEE, vol. 76, no. 5, pp. 578-593 May 1988.

[4] Carl M. Campbell, "Design and specification of cryptographic capabilities," *IEEE Commun. Mag.*,, pp. 15-19, Nov. 1978.

[5] Eric K. Clemons, "Evaluation of strategic investments in information technology," *Communications of the ACM*, vol. 34, no. 1, Jan. 1991.

[6] D. Chaum, J. Evertse, "Cryptanalysis of DES with a reduced number of rounds," in *Advances in Cryptology-Crypto 85*. New York, NY: Springer Verlag, pp. 192-211.

[7] C. A. Deavours, L. Kruth, *Machine Cryptography and Modern Cryptanalysis*. Norwood, MA: Addison-Wesley, 1983.

[8] D. E. R. Denning, *Cryptography and Data Security*. Reading, MA: Addison-Wesley, 1983.

[9] W. Diffie, E. Hellman, "New directions in cryptography," *IEEE Trans. Informat. Theory*, vol. IT-22, pp. 644-654, 1976.

[10] W. Diffie, E. Hellman, "Exhaustive cryptanalysis of the NBS Data Encryption Standard," *Computer*, vol. 10, pp. 74-84, 1977.

[11] W. Diffie, "The first ten years of Public-Key cryptography," in *Proceedings of the IEEE*, vol. 76, no. 5, pp. 560-577 May 1988.

[12] Gilles Garon, Richard Outerbridge, "DES watch: An examination of the sufficiency of Data Encryption Standard for financial institution information security in the 1990's," *ACM Press.*, vol. 9, no. 4, pp. 29-45, 1991.

[13] T. Haight, "Tales From Decrypt," *Network Computing Magazine*, vol. 2, no. 7, pp. 75-83, July 1991.

[14] C. Hammer, "Beale Ciphers," *Cryptologia*, vol. 3, no. 1, pp. 9-15, Jan. 1979.

[15] M. E. Hellman *et al.*, "Results on an initial attempt to cryptanalyze the NBS Data Encryption Standard," Tech. Rep. SEL 76-042, Stanford University, 1976.

[16] Russell Kay, "Infosecurity in the 1990s." *ISPNews*, vol. 2, no. 6, pp. 1, Nov. 1991.

[17] M. Kochanski, "A survey of data insecurity packages," *Cryptologia*, vol. 11, no. 1 pp. 1-15, Jan. 1987.

[18] H. W. Lenstra, Jr. "On the Chor-Rivest Knapsack cryptosystem," *Journal of Cryptology*, vol. 3, no. 3, pp. 149, 1991.

[19] Micheal Merritt, "Theory of Cryptographic Systems: A critique of Crypto Complexity", in *Distributed Computing and Cryptography*. Providence, Rhode Island: American Mathematical Society, and Baltimore, MD: Association for Computing Machinery, 1989.

[20] S. Murphy, "The cryptanalysis of FEAL-4 with 20 chosen plaintexts," *Journal of Cryptology*, vol. 2, no. 3, pp. 145, 1990.

[21] National Research Council, *Computers at Risk, Safe Computing in the Information Age*. Washington, DC: National Academy Press, 1991.

[22] A. M. Odlyzko, "The rise and fall of Knapsack cryptosystems", in *Cryptology and Computational Number Theory*. Providence, Rhode Island: American Mathematical Society, 1989.

[23] William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling, *Numerical Recipes in C, The Art of Scientific Computing*. New York, NY: Cambridge University Press, 1988.

[24] Corey H. Schou, "Hard times: Information security in a changing world," *ISPNews*, vol. 2, no. 5, Sep. 1991.

[25] A. Shamir, "On the security of DES," in *Advances in Cryptology-Crypto 85*. New York, NY: Springer-Verlag, pp. 280-281.

[26] Clifford Stoll, *The Cuckoo's egg, Tracking a spy through the maze of computer espionage*, New York, NY: Pocket Books, 1989.

Additional Readings

E. F. Brickell, J. H. Moore, M. R. Purtill, "Structure in the S-Boxes of the DES (extended abstract), in *Advances in Cryptology-Crypto 86*. New York, NY: Springer Verlag, pp. 3-8.

R. A. Demillo, G. I. Davida, D. P. Dobkin, M. A. Harrison, R. J. Lipton, *Applied Cryptology, Cryptographic Protocols, and Computer Security Models*. Providence, Rhode Island: American Mathematical Society, 1981.

W. F. Ehrsam, S. M. Matayas, C. H. Meyer, W. L. Tuchman, "A cryptographic key management scheme for implementing the Data Encryption Standard," *IBM System Journal*, vol. 17, no. 2, pp. 106-125, 1978.

Peter Fagan, "Experience of Commercial Security Evaluation," in *Proceedings of the 14th National Computer Security Conference*, vol. 1, Oct. 1991.

R. Forre, "Methods and Instruments for Designing S-Boxes," *Journal of Cryptology*, vol. 2, no. 3, pp. 115, 1990.

Zvi Galil, Stuart Haber, Moti Yung, "Security against Chosen-Ciphertext Attack", in *Distributed Computing and Cryptography*. Providence, Rhode Island: American Mathematical Society, and Baltimore, MD: Association for Computing Machinery,1989.

W. Mark Goode, "Crypto standards: A thousand points of connection?," *ISPNews*, vol. 2, no. 5, pp. 43, Sep. 1991.

B. S. Kaliski, R. L. Rivest, A. T. Sherman, "Is the Data Encryption Standard a group (result of cycling experiments on DES)," *J. Cryptology*, vol. 1, no. 1, pp. 3-36, 1988.

Neal Koblitz, *A Course in Number Theory and Cryptography*. New York, NY: Springer-Verlag, 1987.

A. G. Konheim, *Cryptography, A Primer*. New York, NY: Wiley, 1981.

Stephen M. Lipton, Stephen M. Matyas, "Making the Digital Signature legal and safeguarded," *Data Communications Magazine*, pp. 41-52, Feb. 1978.

James L. Massey, "An introduction to contemporary Cryptology," in *Proceedings of the IEEE*, vol. 76, no. 5, pp. 533-549 May 1988.

Terry Mayfield, Stephen R. Welke, John M. Boone, Catherine W. McDonald, "A framework for advancing integrity standardization," in *Proceedings of the 14th National Computer Security Conference*, vol. 1, Oct. 1991.

C. H. Meyer, S. M. Matyas, *Cryptography: A New Dimension in Computer Data Security*. New York, NY: Wiley, 1982.

J. H. Moore, G. J. Simmons, "Cycle structure of the DES with weak and semiweak keys," in *Advances in Cryptology-Crypto 86*. New York, NY: Springer-Verlag, pp. 9-32.

Robert Morris, N. J. A. Sloane, A. D. Wyner, "Assessment of the National Bureau of Standards proposed Federal Data Encryption Standard," *Cryptologia*, vol. 1, no. 3, pp. 281-291, 1990.

D. B. Newman, Jr., R. L. Pickholtz, "Cryptography in the private sector," *IEEE Commun. Mag.*, vol. 24, pp. 7-10, Aug. 1986.

K. Nishimura, M. Sibuya, "Probability to meet in the middle,"*Journal of Cryptology*, vol. 2, no. 1, pp. 13, 1990.

NIST, "Data Encryption Standard (DES)," in *FIPS publication 46*, National Technical Information Service, Springfield, VA, Apr. 1977.

NIST, "Guidelines for implementing and using the NBS Data Encryption Standard," in *FIPS publication 74* , National Technical Information Service, Springfield, VA, Apr. 1981.

NIST, "DES modes of operation," in *FIPS publication 81*, National Technical Information Service, Springfield, VA, Dec. 1981.

Donn Parker, "Restating the Foundation of Information Security," in *Proceedings of the 14th National Computer Security Conference*, vol. 2, Oct. 1991.

Charles R. Pierce, "Standardized Certification," in *Proceedings of the 14th National Computer Security Conference*, vol. 2, Oct. 1991.

Charles R. Pierce, "Toward certification, a survey of three methodologies," in *Proceedings of the 14th National Computer Security Conference*, vol. 2, Oct. 1991.

Arto Salomaa, *Public-Key Cryptography*. Berlin, Heidelberg, New York, NY: Springer-Verlag, 1990.

Miles E. Smid, Dennis K. Branstand, "The Data Encryption Standard: Past and future," in *Proceedings of the IEEE*, vol. 76, no. 5, pp. 550-559, May 1988.

RE-USE OF EVALUATION RESULTS

Jonathan D. Smith

Admiral Management Services Ltd.
Commercial Licensed Evaluation Facility
Kings Court, 91-93 High Street
Camberley, Surrey, GU15 3RN
ENGLAND

ABSTRACT

As evaluated products become more widely available and better focused at purchaser's requirements, re-use of previous evaluation results becomes an increasing priority. This is particularly true if 'monolithic' evaluations are to be avoided for systems that incorporate previously evaluated products. For systems that incorporate previously evaluated products, evaluation becomes more cost effective if it is possible to benefit from the results of the previous product evaluations. This discussion paper addresses this issue through a discussion of the Information Technology Security Evaluation Criteria [ITSEC] and the process of composition of components. It then proposes a possible way forward for the re-use of evaluation results. The paper assumes a working knowledge of the ITSEC.

INTRODUCTION

A key issue in the development and evaluation of secure systems and products (referred to by ITSEC as Targets Of Evaluation - "TOEs") is: how can the results of previous evaluations (in terms of evaluation level, functionality and strength of mechanisms) be re-used in another development/evaluation? This issue is difficult to resolve and, in Europe to date, has been left to the discretion of the certifiers and accreditors of TOEs.

This paper introduces a number of aspects of ITSEC that are relevant to re-use of evaluation results, including:

- *Flexibility of approach*: Evaluation is made more flexible by splitting the functionality of a TOE from the confidence held in a TOE to meet its security target (for instance limited functionality can now be evaluated to a high degree of confidence). However, the cost of this approach is that a diverse range of functionality and confidence has to be taken into account when evaluation results are re-used

- *The difference between effectiveness and correctness*: The ITSEC identifies these two properties of a TOE as a starting point for the evaluation criteria, thereby forcing them to be considered explicitly during any treatment of the re-use issue

- *The assumption that systems and products can always be considered in the same way*: The veracity of this assumption cannot be guaranteed when the re-use problem is addressed.

The paper then discusses composition of components. A number of options to address the re-use issue are proposed. The implications of each option are examined with reference to a simple example. Those options considered are:

- *Assurance profiles*, as permitted by the ITSEC

- *Mixing rules*, for generating a single functionality statement/evaluation level from assurance profiles

- *Expanding each relevant criterion of the ITSEC.*

Finally, conclusions are drawn on combining the above options to enable a way forward for the re-use of evaluation results.

THE ITSEC FRAMEWORK

Introduction

This section introduces the relevant aspects of the ITSEC in order to provide a foundation for the discussion of composition of components. This paper assumes a working knowledge of the terminology and concepts used in [ITSEC] beyond the following paragraphs.

Development Model

The assumed ITSEC development model is given in Figure 1.

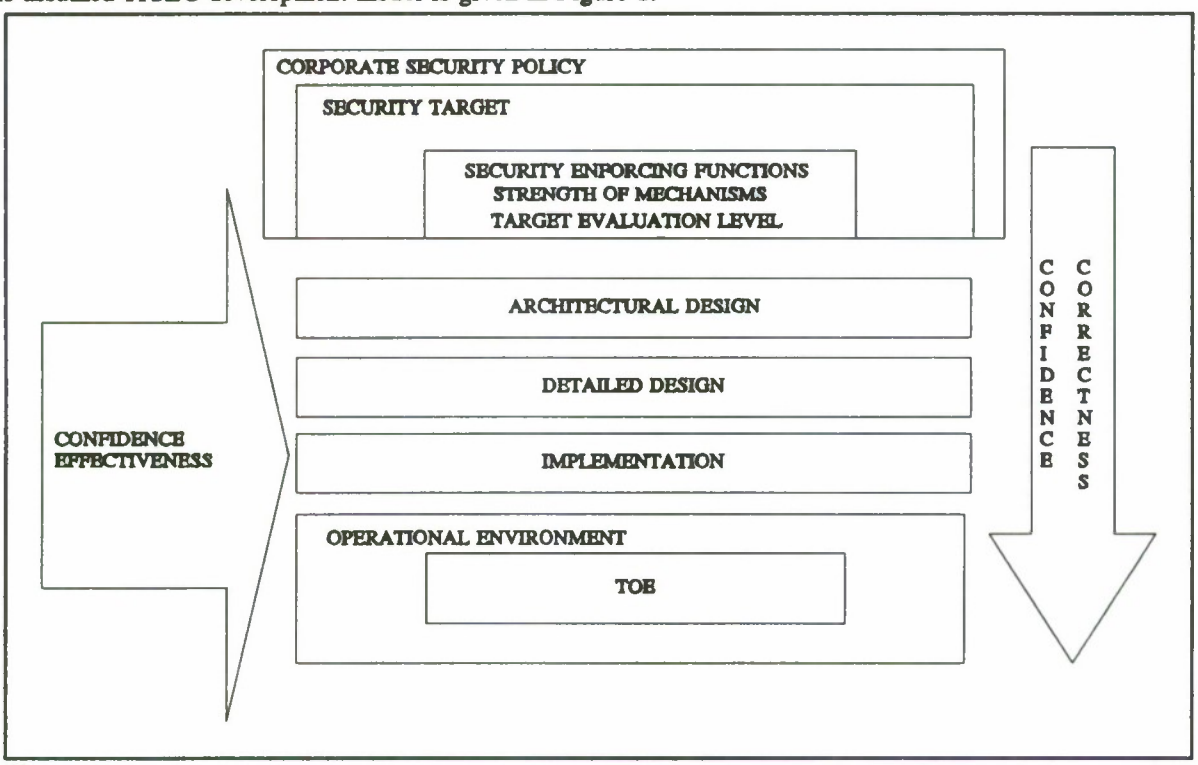


Figure 1

Notice that the figure assumes a single layer of Detailed Design. The ITSEC, however, identifies a design hierarchy within the Detailed Design. The assumption has been made for the sake of simplifying the figure and the discussion that follows.

Security enforcing functionality is first identified in a security target (for instance as a set of security enforcing functions). A security enforcing function is identified in order either to counter a specific threat (more likely for a system) or to meet a specific security objective (more likely for a product).

A TOE is made up of components which may themselves be made up of components. The ITSEC defines a component as an identifiable and self-contained portion of a TOE, and identifies a 'basic component' at the lowest level of design. A TOE's security target can therefore be refined into a number of security enforcing, security relevant and other components. It should be noted that a security enforcing component may be refined further into security enforcing and security relevant components.

The Architectural Design identifies the first allocation of functions to components. The Architectural Design therefore identifies the *separation* between security enforcing and other functionality.

The Detailed Design (hierarchy) identifies further components. Security irrelevant components can be identified by ensuring that, at any design phase, whatever functionality they implement, and whatever behaviour they exhibit at their interfaces, providing that the security enforcing and security relevant components operate correctly, the TOE continues to operate securely.

Security relevant components can be identified as those components that are not security irrelevant components but whose abstraction is *not* defined in the security target (as one or more security enforcing functions).

Security enforcing components are components that are not security irrelevant components and whose abstraction *is* identified in the security target.

Flexibility of Approach

Traditionally, a key component of a secure product or a system has been the reference monitor. This component provides the foundation for the Trusted Computing Base as defined in [TCSEC]. The reference monitor concept has been 'loosened' in the ITSEC to allow secure products (and systems) to encompass more than just confidentiality (and mediation between subjects and objects). However, in order to compensate for this more flexible approach, separation requirements on the Architectural Design are made explicitly by the ITSEC. Therefore, for TOEs to implement a correct access control policy at higher evaluation levels the requirements of the reference monitor concept still hold; i.e. architectural separation of the component that implements the access control requirements is still required.

However, if access control requirements are not addressed through the use of a component that mediates accesses to objects by subjects, which is both tamper resistant and small enough to be subject to analysis and (complete) tests, then the effectiveness of the developer's proposed solution has to be considered.

Effectiveness and Correctness

Correctness is concerned with correct refinement of security enforcing functions through the representations of a TOE. Effectiveness is concerned with the ability of a TOE to meet its security objectives or counter threats when the TOE is considered as a whole.

Effectiveness and correctness are complementary in the ITSEC framework; sometimes covert channels are an aspect of correctness (if requirements about them are specified in the security target) and sometimes effectiveness.

Systems and Products

Broadly, systems and products can be treated in similar ways for evaluation purposes, and therefore criteria can be written for TOEs as either systems or products. However, there are fundamental differences between systems and products:

- A *system* in its broadest IT context is part of an organisation which performs particular functions and which includes both a TOE and its environment (users etc.) working together

- A system, in TOE terms, therefore includes an operational environment (for instance secure operating procedures), and the threats to the system can be countered through the use of physical, procedural or personnel measures, as well as through electronic countermeasures (the first description of which forms the set of security enforcing functions specified in the security target)

- A *product*, by definition, has no specific system or organisation of which it is a part. In other words, the environment for a product is not as well defined as for a system -assumptions about threats (and even assets) will have to be made in a 'product rationale' (probably through the use of security objectives).

To summarise, a system is a component of an organisation, itself made up of components made up of components etc. A product may be made up of components. A product, when re-used in a TOE (at the Architectural or Detailed Design representations) will be incorporated into that TOE as a component.

When a product is purchased for use within a system the purchaser 'accredits' the product. For systems, a system is usually accredited by a third party to ensure that the information that the system processes will remain secure.

COMPOSITION

Introduction

In order to discuss re-use of a component a description of both a component and composition is required. A component is an identifiable and self-contained portion of a TOE. Any component can be described by a statement of its major attributes:

- Its functionality

- The external interface which it offers its environment

- The assumptions that it makes about its environment in order for it to work correctly.

This is true for the lowest level of component (basic component) or a component which is a composition of two or more components (a target component). The definition of a component is therefore recursive. This is an important point because a component can be broken down into further components until a basic component is finally reached. An example is presented in Figure 2.

Composition is the 'bottom up' process of combining components in order to meet the requirement on the ultimate target component (in the case of evaluation the TOE). A target component could therefore be any of:

- A system

- A product

- A component of a product or system.

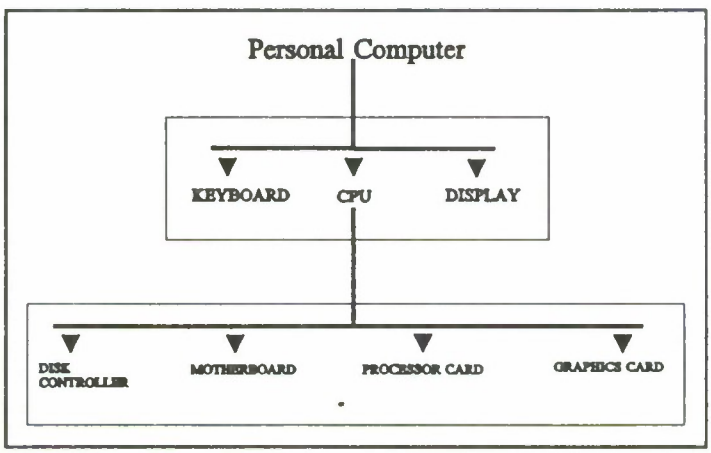


Figure 2

Relevance to Development and Evaluation

To develop a target component (for instance a TOE) the three major attributes of the target component will be specified in the overall design for the target component. These specifications form the predicates upon the result of the composition process that have to hold in order that the target component is implemented both correctly and effectively.

Notice that the definition of a component is, in essence, a re-statement of a product's security target: the ITSEC has already assumed that a product will be designed as a component.

The statement of functionality for the target component can specify either security enforcing, security relevant, or security irrelevant functions. The functionality statement can range from a complete security policy to a description of one necessary property of a component.

The functionality which a component's interface provides can be either services to be exploited by another component or services delivered to a user. When components are composed the interface can therefore be a consumer interface or a producer interface. The precision of description of the interface and the functionality that it delivers depends on the component's original target evaluation level.

There are two issues to consider regarding the assumptions that the component makes about its environment:

- Firstly, the assumptions made regarding the non-IT services it relies on in order to function correctly and effectively (e.g. for confidentiality requirements that an attacker cannot physically access/change hardware - a secure environment encapsulates the TOE)
- Secondly, assumptions made about the IT services (external to the component) it uses (i.e. requires in order to function correctly and effectively) through its interface.

It should be noted that the greater the level of self-protection (for instance tamper-proofing) built into a component, the fewer assumptions about its environment need to be made for it. Any assumptions need to be stated as part of the component's specification.

This discussion has highlighted an obvious issue: *at what stage in the development process is re-use of a component relevant to evaluation?* It is asserted here that components can only be re-used at the Architectural and Detailed Design levels. A component cannot be re-used as part of the implementation of a TOE without having been introduced at a higher design representation: to re-use a component in the implementation, its intended use must have been specified in at least the Detailed Design.

However, there is a side issue here: a security target may identify specific components to be re-used (for instance a specific hardware platform). If this is the case then the security target has already started to specify the Architectural Design for the TOE. Further, a security target also specifies a strength of mechanisms rating. A mechanism can be considered to be specified at the Detailed Design level. Therefore the security target has already started to specify the Detailed Design for the TOE.

Problem Summary

The discussion above has treated components as 'black boxes'. The primary effect of introducing an additional attribute of a target component, that of its target evaluation level, is that the evidence required to meet the evaluation level reveals to the evaluator some of the target component's functionality, interface and environmental assumptions. From this point onwards this paper assumes that the target component is the TOE to be evaluated.

The TOE will have been the result of a composition process. The evaluation level that the TOE is aimed at will therefore also reveal to the evaluator some of the internal functionality, interface and environmental assumptions for each of the components composed into the TOE. This is necessary for confidence in the TOE to be obtained.

In summary, for secure developments the following must hold:

- The TOE, and therefore each component that it contains, must be correctly implemented - this means that the requirements on the TOE must be upheld correctly by the sum of the functionality provided by each component
- All interfacing between components must 'match'
- The assumptions that each component makes about its environment must be upheld by the TOE
- Any known vulnerabilities in any of the components must not become exploitable in the TOE.

The following problems with demonstrating the above often occur:

- The functionality of the TOE is not fully provided by the functionality of the components - often components are re-used in different ways for different purposes than they were originally intended (particularly true when non-evaluated components are re-used in a TOE that is to be evaluated) and therefore bespoke components are needed

- Interfacing between components does not match because components only use subsets of the interfaces provided by other components or the interface descriptions are specified at different granularities
- Environmental assumptions for the TOE are different to those made for individual components when the components were designed
- The confidence in the secure operation of each component (and therefore the confidence in the absence of exploitable vulnerabilities in the TOE) is different - the TOE has an assurance profile.

It should therefore be clear that some form of evaluation is always required for TOEs even when all components (products) that are re-used have been previously evaluated to the same level as that specified for the TOE.

GENERIC GUIDANCE BASED ON DISCUSSION OF COMPOSITION

Generally, the results of correctness assessment can be re-used for the internal functionality and interface of a component.

Effectiveness is concerned with compositions and addresses interfacing to the user, environmental assumptions and vulnerabilities within and between components. It should be noted that a component may contain vulnerabilities which become exploitable only after its incorporation into the TOE. Therefore effectiveness assessment must always be re-performed whenever composition of components is undertaken.

In terms of the effectiveness of the composition of components:

- *Suitability analysis* has to establish whether the security enforcing functions for the TOE can be upheld by the individual statements of functionality for each component of the TOE
- *Binding analysis* must then be performed on the TOE to ensure that the interfacing and assumptions of each component are upheld
- *Construction vulnerability analysis* must be performed on the TOE to ensure that vulnerabilities within each component do not become exploitable as a result of the composition - the vulnerabilities of the components must be listed in the report from their previous evaluation
- *Ease of use analysis* must be performed on the TOE.

OPTIONS

Introduction

A variety of options for accommodating re-use of evaluation results exist. Those considered here are:

- *Assurance profiles*
- *Mixing rules* for generating a single functionality statement/evaluation level from confidence profiles
- *Expanding each relevant criterion of the ITSEC*
- A combination of the above.

Assurance Profiles

Assurance profiles can be used (the ITSEC does make reference to these in Chapter 1). Assurance profiles allow different sets of security enforcing functions to be evaluated to different evaluation levels through the use of multiple security targets.

It is postulated here that what is actually required is:

- An overall system security target - which identifies one required strength of mechanisms rating, the security enforcing functions together with their allocation to components, and the target evaluation levels for each component (the target evaluation levels for each component will be determined based on a threat distribution assessment)

- Individual security targets for each component - identifying the relevant security enforcing functions etc. and the assumptions regarding the environment of the component made for the component to work securely.

To re-use the results of a previous evaluation of a component, the developer would then have to produce a rationale as to how the functionality expected from the previously evaluated component is consistent with the previously evaluated functionality: functionality of the component that was previously evaluated must conform to its new predicates specified in its new security target.

This is clearly impractical for products without a clear statement of 'mixing rules' (see below), for a product may have a rating of [[E5,F-B2],[E2,F-**]] which would be very difficult to incorporate sensibly into a system. However, in the absence of any prescriptive guidance, an assurance profile may be acceptable to system accreditors.

Mixing Rules

Ideally, rules are required for composing components such as a security enforcing e.g. [E5,F-B2] component with a security relevant e.g. [E3,F-**] component to produce a [E5,F-B2] system. However, the discussion above demonstrates that the internal functionality of the security relevant [E3,F-**] component may result in an invalid composition that fails to meet the [E5, F-B2] requirement without further work being performed by the developer (and also see [TDI]).

However, this form of composition may be possible given the next option - extensions to the criteria.

Expansion of the Relevant ITSEC Criteria

A key aspect of the discussion above is the fact that components can be re-used *only* at the Architectural and Detailed Design representations. This results in a limited impact of re-use on the criteria that can be resolved by updating the individual criteria.

The ITSEC criteria for Architectural and Detailed Design would then need to include rules such as:

If a security enforcing component is specified then
    If previously evaluated to > = overall target evaluation level then
        Previous correctness assessment results are valid
    If previously evaluated < overall target evaluation level then
        Do additional correctness assessment actions, for overall target level, beyond previous evaluation level

{NB: Additional correctness actions include those for development environment etc.}

If a security relevant component is specified then
    If previously evaluated to > = [overall target evaluation level-1] then
        Previous correctness assessment results are valid
    If previously evaluated to < [overall target level-1] then
        Do additional correctness assessment actions, for [overall target level - 1], beyond previous evaluation level

Do effectiveness assessment at overall target evaluation level or highest evaluation level specified for a component

Of course, it may be the case that the additional evidence required for correctness (beyond the previous evaluation level for the component) cannot be supplied. Then the proposed TOE would have to fail evaluation at its overall target evaluation level.

It should be noted that the suggested update of the criteria in the way that security relevant components are evaluated has an impact on all evaluations - not just those where re-use is an issue.

These are just examples to highlight what may be possible (and in fact relate to the inheritance aspect of decomposition identified in safety critical standards such as DEF STAN 00-56 [DEF-STAN]).

Re-use of Strength Ratings

Strength ratings can be re-used according to the strict rule that the rating of high, medium or low of the individual components must be greater or equal to the strength rating claimed for the TOE.

AN EXAMPLE COMPOSITION

Introduction

Consider the case of an information management system to be implemented using a bespoke application e.g. bespoke application software (including a Human Computer Interface - "HCI") together with two further components:

- An 'off the shelf' database management system (COMP1)

- An underlying operating system/hardware platform (COMP2).

The new system is targeted at an evaluation level of E3. The security functionality required is based upon the F-B1 functionality class, with additional availability and integrity requirements. In this example of composition many design options are available to the developer.

Accounting And Audit Design Options

Some of the design options for meeting the accounting and audit requirements for this example are now considered further. The design options considered here are:

- Exploit the operating system services available for both accounting and audit purposes (Option 1)

- Exploit the operating system accounting services and develop a bespoke audit tool (Option 2)

- Exploit the operating system accounting services and integrate a commercially available audit product (Option 3).

Option 1

The accounting and audit components of the operating system are to be re-used in the system. These are classed as security enforcing components for the system and hence must be provided by an operating system previously evaluated to E3. (Note that if requirements on these components were not specified in the security target for the system, then they may have been classed as security relevant or even security irrelevant components.)

The set of predicates on the accounting and audit components are specified in the information management system security target. The previously evaluated operating system security target must therefore be consulted to ensure that the internal functionality will implement the predicates assumed for the components.

The previous Evaluation Technical Report for the operating system will have to be consulted to ensure that there are no vulnerabilities that could become exploitable in some way in the new system context (as part of the new system's construction vulnerability analysis).

During detailed design the interfacing to the accounting and audit facilities will have to be addressed. For instance, the HCI interface will have to be determined.

In summary, the correctness of the previous evaluation results can be assumed. However, the effectiveness assessment will address:

- The suitability of the accounting and audit components of the operating system through a new suitability analysis (which examines the original operating system's security target)
- The binding of the accounting and audit services with the actual system implementation (and the HCI in particular)
- Any potential system construction vulnerabilities introduced - including any raised during the previous evaluation
- The ease of use of the new system implementation.

Option 2

The major difference from Option 1 in using a bespoke auditing application is that the implementation of the audit trail will form part of the evaluation. The auditing application will have to be targeted at E3 and developed and evaluated accordingly.

Option 3

The major difference of this scenario from the previous two options is that the auditing product will either have to have been previously evaluated or will have to form part of the development/evaluation at E3 - all the deliverables to the evaluation associated with E3 must be available to the evaluators for the 'monolithic' evaluation.

If the audit product had previously been evaluated to E3 then the development/evaluation would proceed as in Option 1 above. If the developer intended to use an auditing package that was previously evaluated to E2, then the developer could:

- Arrange for the additional evidence required for E3 to be delivered to the evaluators and evaluated
- Convince the certifier/accreditor to require only E2 for the audit component, thereby allowing the correctness of the previous evaluation results to be re-used, and the system to be 'profiled'
- If possible, avoid stating the audit requirements in the system security target, thereby transforming the audit component into a security relevant or irrelevant component.

However, given the suggested updates to the ITSEC, if the accreditors accepted that the audit component was security relevant, then the developer could re-use the audit product at E2 totally legitimately; additional work to achieve E3 would not be required.

WAY FORWARD

Introduction

As a result of the previous discussions, this section briefly proposes:

- How to handle re-use in the interim

- A longer term approach to resolving the re-use issue.

Preliminary Approach

Assurance profiles for systems are required in the short term - accreditors will have to assess the threat/countermeasure distribution on the system on a case by case basis. Systems may be given profiled ratings such as [[E3, F-B1][E2, F-**]...].

Development and evaluation of the overall system should always include effectiveness assessment commensurate with the highest target evaluation level specified for a component.

Products should be targeted at a single evaluation level in order that sensible re-use of their evaluation results can be made when they are incorporated into systems (products could be decomposed to a level such that they can be targeted at one evaluation level, if necessary).

Longer Term Approach

The Architectural and Detailed Design criteria of the ITSEC should be updated to take composition of components into account explicitly. In the light of the expanded criteria, and further utilisation of the discussion of composition, it may be possible to derive mixing rules to allow an [E3, F-B1] component to be combined with an [E2, F-**] component to produce an overall rating of [E3, F-B1].

Practical research is required, particularly in the area of how threat distribution might be used to enable valid mixing rules to be formulated.

SUMMARY

Re-use of evaluation results is a hard problem. The above discussion has highlighted a potential way forward that requires significant work and input from developers, accreditors, certifiers and evaluators.

Without resolution of the re-use issue via the provision of clear guidance product developers will find it difficult to assess the marketability of their product at a particular target evaluation level.

REFERENCES

ITSEC — Information Technology Security Evaluation Criteria, Harmonised Criteria of France, Germany, the Netherlands, United Kingdom, Version 1.2, June 1991.

TCSEC — Department of Defense Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, Department of Defense, December 1985.

TDI — Trusted Database Management System Interpretation, NCSC, April 1991.

DEF-STAN — Hazard Analysis and Safety Classification of the Computer and Programmable Electronic System Elements of Defence Equipment, Ministry of Defence, April 1991.

Since 1985 Admiral has been involved with the Communications - Electronics Security Group (CESG) in the development and application of standards and procedures for security evaluation. In 1985 the Company commenced operation of the lead Evaluation Facility for CESG, and in 1991 the Evaluation Facility was accredited by the National Measurement Accreditation Service as a Commercial Licensed Evaluation Facility (CLEF).

Jonathan Smith is a member of the Admiral CLEF, and has been a contributor to the Information Technology Security Evaluation Manual and UK IT Security Evaluation And Certification Scheme Publications.

RISK MANAGEMENT OF COMPLEX NETWORKS

by Richard Cox and Dr. Michael O'Neill
CTA Incorporated, 7150 Campus Drive, Suite 100, Colorado Springs CO 80920-3178
and Lt Col William Price, HQ AFSPACECOM/LKXS, Peterson AFB CO 80914-5001

ABSTRACT

Regarding communications-computer systems security, current Department of Defense (DoD) guidance readily applies to standalone systems or simple networks. However, these documents do not apply easily to complex evolving networks -- especially heterogeneous conglomerates which have existed for several years and are evolving to meet operational requirements.

This paper describes a methodology for risk management of complex networks. There are three primary tasks: determining a security policy, constructing a security architecture, and developing an accreditation strategy.

INTRODUCTION

According to DoD policy, all communication-computer systems are subject to "risk management," a process to identify, evaluate, and reduce risks and vulnerabilities. The ultimate goal of the risk management process is to operate at an acceptable level of risk. This results in accreditation (written approval to operate) from a Designated Approving Authority (DAA).

A non-trivial problem for many DAAs is how to apply current DoD guidance in accrediting complex evolving networks. Publications such as DoD Directive 5200.28 [1], the "Orange Book" [2], and the "Red Book" [3] apply best to systems such as standalone mainframes or simple networks; these publications do not directly address complex networks. Combining standalone systems into a network invariably results in security issues (such as the cascading problem) which are unique to networks. Recognizing this, the National Research Council stated, "there is also a need to address broader system security concerns in a manner that recognizes the heterogeneity of integrated or conglomerate systems." [4]

Complex networks generally have the following characteristics:

- There are at least two interconnected networks. In some cases, there are many interconnected systems or networks performing critical missions.

- There are many organizations involved in development and acquisition of components, so there are multiple DAAs. There are usually two or more network DAAs.

- It is a multilevel network; computers or network nodes process different levels of classified information while users have different clearances and need-to-know restrictions.

- The network has evolved over a period of years, and it continues to evolve. Many of the systems predate modern computer security features or capabilities, and subsequent evolutions have not explicitly followed these security practices. However, total upgrades or replacements are neither operationally nor economically feasible.

- The operating command uses and controls the network, and although some components may have individual approval to operate, the overall network does not have accreditation.

- There is no overall network security policy.

Copyright 1992 CTA Incorporated

For such networks, the risk management process can be extremely complicated and resource intensive. This paper proposes a risk management methodology which could be followed to accredit any complex network. A key element of the methodology is reducing complexity to a level of abstraction (information reduction) which allows meaningful application of current DoD guidance while meeting operational requirements. In the cybernetic systems disciplines this is often referred to as the "Cones of Resolution," or trying to comprehend the logic of the basic elements as well as the relationships among the elements. [5] Recent literature refers to this process as "transition engineering methodology" -- reducing complexity by using data reduction and data abstraction as a means to describe and analyze large complex evolving systems. [6]

Although the risk management methodology can be described as three tasks, they are not discrete phases, but parallel and interwoven activities. For example, security policy usually drives the security architecture; but complex networks which have evolved for years often do not have an overall security policy. It might be necessary to "reverse engineer" a security solution for the network, so architectural analysis could precede policy determination to assist in understanding the complexities of the system. For any complex network, the three tasks should evolve and iterate concurrently. Although there are several steps within each task, they are identified according to the primary goal of that task. They are:

- Task I: Determine a security policy.

- Task II: Construct a security architecture.

- Task III: Develop an accreditation strategy.

PRELIMINARY RESEARCH

Before Task I of the methodology, do preliminary research to determine basic information about the network:

First, determine network boundaries. Since the goal is to accredit a complex network, determine boundaries to clearly discriminate between what is in the network and what is in the environment; between what will be and what will not be accredited. Identify "core" systems whose primary mission is to process data for the network, and "affiliated" systems whose primary mission is something other than processing data for the network.

Second, review network documentation. Having determined the boundaries of the network, collect and review all applicable documentation to:

- Identify the overall mission and functions of the network.

- Identify how the network is managed and controlled.

- Identify the networks, sub-networks, and communications-computer system components to be accredited as part of the overall network. These include (but are not limited to) data sources, communications paths from the data sources, processing entities, and communications paths to the users.

Third, identify the overall network DAA. This is the top-level DAA with authority to grant approval to operate for the entire "network of networks."

TASK I -- SECURITY POLICY

Because most complex networks have evolved for years, they often have significant weaknesses by current security standards. Establishing an overall security policy suggests solutions to security problems and provides guidelines for future evolution. The security policy should accommodate the existing system as much as possible to minimize the cost of changing the network. Task I produces a Network Security Policy (NSP) which should address at least seven areas:

(1) Network Security Objectives. Identify primary and secondary security objectives. The primary objectives will include confidentiality, integrity, and availability.

(2) Roles and Responsibilities. Establish specific duties and tasks for all personnel with network interfaces. These include (a) DAAs at all levels; (b) MAJCOM and Base Level Communications-Computer Security Officers (CSOs), MAJCOM and Base Computer Systems Security Managers (MCSSMs and BCSSMs); (c) Network Managers (NMs) and Network Security Officers (NSOs); (d) Computer Systems Managers (CSMs) and Computer Systems Security Officers (CSSOs); (e) Functional Area Managers (FAMs) and Terminal Area Security Officers (TASOs); (f) systems analysts, programmers, and software support personnel; (g) system administrators; and (h) computer/network users.

(3) Life Cycle Management. Identify life cycle phases for network additions, deletions, and modifications, and especially for new systems being developed to interface with the network. Explain how DAAs will accredit and reaccredit systems throughout their life cycle. Emphasize the need to continually monitor the network to ensure security measures remain effective after system changes.

(4) Network Security Measures. Specify policies and procedures regarding:

- Physical security, which involves protection and survivability for personnel and equipment; protection against intentional human threats such as theft, sabotage, and espionage; and environmental security.

- Procedural security, which protects against unintentional human threats such as inadvertent blunders, improper maintenance, etc.

- Personnel security, involving clearance (including access to categories or compartments) and need to know.

- Information security (INFOSEC), which includes procedures for handling classified and sensitive unclassified information; magnetic remanence; fraud, waste, and abuse, etc.

- Communication security (COMSEC) to protect secure and sensitive communications.

- Emanations security (TEMPEST) to prevent exploitation of electronic signal radiations.

- Operations security (OPSEC) to identify, control, and protect evidence of the planning and execution of sensitive activities.

- Trusted systems security, which includes trusted computing base (TCB) classes, modes of operation, identification and authentication, discretionary and mandatory access control, object reuse, audit trails, labels, trusted paths, and documentation requirements.

- Hardware security, which involves nonvolatile storage media, peripheral security, maintenance activities, periods processing, and firmware.

- Software security, which involves evaluated and non-evaluated products; user-developed software; public domain software, freeware, and shareware; software development, testing, and debugging; security software; job control language; configuration management; trusted software, data base management systems (DBMS); maintenance activities; and malicious logic.

- Integrity measures such as device identification, message management, protocols, and integrity checks.

- Other security considerations such as interface policies and resource allocation policies.

(5) Contingency and Emergency Plans. Establish criteria for developing and testing contingency and emergency plans, especially policies for making and storing backups.

(6) Education, Training, and Awareness. Identify policies and procedures for all aspects of security training, including initial and recurring training for all personnel, specialized training requirements for personnel in key security positions, and minimum standards for NSO-developed training.

(7) Incident and Vulnerability Reporting. Establish policies and procedures for identifying and reporting incidents and vulnerabilities.

Policy development must focus on specifically tailoring DoD guidance to the network. The Orange Book uses abstract terms such as subjects, objects, groups of subjects, need-to-know, security labels, discretionary access controls, and mandatory access controls. The policy should describe what these abstract terms mean in the context of the network. Key decisions might be whether network nodes are subjects, or whether users or processes residing on the nodes are processes. The former view allows the policy to operate at a higher, more abstract level, while the latter leads to a more complex policy. If the former view were feasible, the developer of a particular node would refine the policy for the particular node. The refined policy would have to address the users and processes operating on the node.

The policy should address specific issues such as the range of security labels which the network must accommodate; specific objects having security labels; how security labels are determined for objects; the various users, communities, groups of users, and actions which are subject to discretionary or mandatory access controls.

TASK II -- ARCHITECTURE

Task II has three steps: construct the general architecture, construct the security architecture (a subset of the general architecture), and identify security architecture issues. The primary goal of this task is constructing the security architecture.

Step 1: Construct the general architecture. From high-level perspectives, it is possible to construct many different general models of the network. The methodology suggests constructing general architectures (including simple models or diagrams) from at least two points of view:

- Mission architecture. Divide the primary mission into mission components, then determine which centers and information processing centers support each component. Finally, determine which elements of the network support the centers to accomplish their missions. (Some elements may support more than one mission.) If the network has a command and control structure, identify the flow of mission command and control from the highest agencies controlling the network to the lowest agencies contributing to the network.

- Communications-computer system architecture. Identify data flow from individual sources, through various communications channels, to processing entities such as correlation centers, through other communications channels, to the network users.

Step 2: Construct the security architecture. While general architectures can be done at a high level, the security architecture must be very detailed. It must identify general security facts, assess the network's ability to meet primary security objectives, and describe protection mechanisms used by the network.

First, identify general security facts about the entire network:

- Highest and lowest classifications processed (including categories and compartments), or types of sensitive unclassified being processed.

- Minimum and maximum user clearances and restrictions (i.e., user limitations based on clearance, access to categories and compartments, or need to know).

- All security modes of operation (dedicated, system high, partitioned, or multilevel) used throughout the network.

- All TCB classes (ranging from class D to class A1) used throughout the network.

Second, assess how the network achieves these three security objectives:

- How does the network protect classified or sensitive unclassified data from unauthorized disclosure (confidentiality)?

- How does the network ensure system integrity (the ability to function unimpaired, free from deliberate or inadvertent unauthorized manipulation) and data integrity (i.e., data correctly represents information, and authorized users and network processors handle and manipulate the data properly)?

- How does the network provide both assurance of service (for authorized users) and denial of service (to unauthorized users)?

Third, specifically identify and describe mechanisms which protect against common threats and vulnerabilities to the network. Be sure to cover all the disciplines mentioned in the NSP, including physical security, procedural security, personnel security, INFOSEC, COMSEC, TEMPEST, OPSEC, trusted systems security, hardware and software security, integrity mechanisms, and other security measures.

Step 3: Identify and document security architecture issues uncovered during network investigation and analysis. Make recommendations to resolve these issues.

TASK III -- ACCREDITATION STRATEGY

Core systems (whose primary mission is to process data for the network), follow a particular process to obtain "approval to operate," while affiliated systems (whose primary mission is something other than processing data for the network), follow different procedures to obtain "approval to connect."

For core systems, Task III has four steps: establish a security management structure, develop risk analysis and certification procedures for security personnel, establish standard procedures for component DAAs to grant approval to operate (for core systems) or approval to connect (for affiliated systems), and establish a mechanism for the overall network DAA to accredit the entire network.

To illustrate this process, Figure 1 shows a relatively simple "network of networks." For simplicity, there are only nine Central Processing Units (CPUs) grouped into three networks, plus the overall "network of networks." Because the CPUs are assumed to be geographically separated, there are nine DAAs for the nine CPUs, three DAAs for the three networks, and one network DAA for the "network of networks" -- a total of 13 DAAs in this simple example.

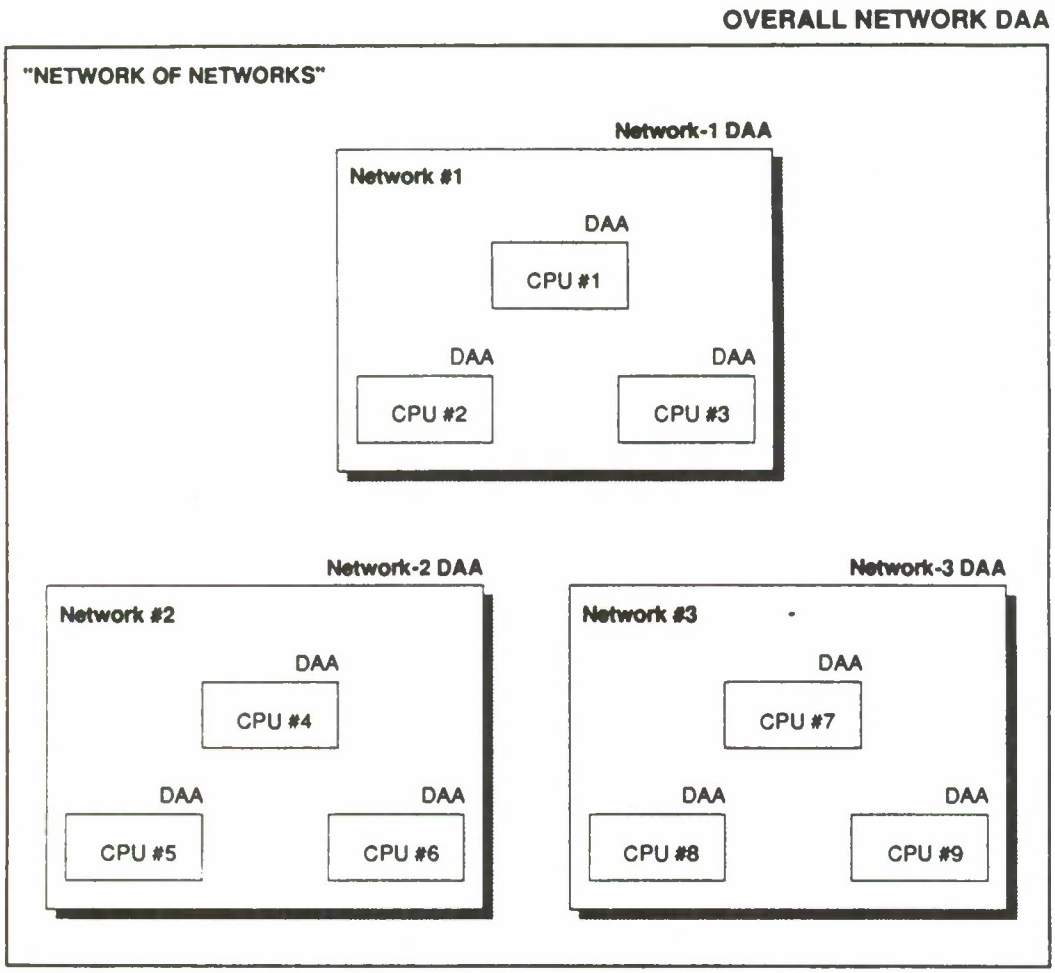


Figure 1. A Simple "Network of Networks"

Step 1: Establish a security management structure to implement and enforce the NSP. Begin by appointing the overall network DAA, then (throughout the entire network) ensure the appointment of component DAAs, network managers (NMs), computer system managers (CSMs), network security officers (NSOs), and computer system security officers (CSSOs). Finally, establish security working groups to represent the various interests and organizations within the network. These groups are forums to identify and resolve security relevant issues.

Because there are multiple DAAs, approval is an iterative process, repeated at least once by each of the component DAAs under the overall network DAA. Figure 2 shows a "pyramid" of DAAs with the overall network DAA at the top and four levels of component DAAs underneath.

Approval starts at the bottom, when "Level 4" DAAs approve their systems and forward paperwork to their "Level 3" DAAs. Then the "Level 3" DAAs approve their systems and forward paperwork to their "Level 2" DAAs. The process continues until the "Overall Network DAA" receives paperwork from the "Level 1" DAAs and accredits the entire network. This iterative process means that steps 2 and 3 (risk analysis and certification, and DAA approval) are repeated many times.

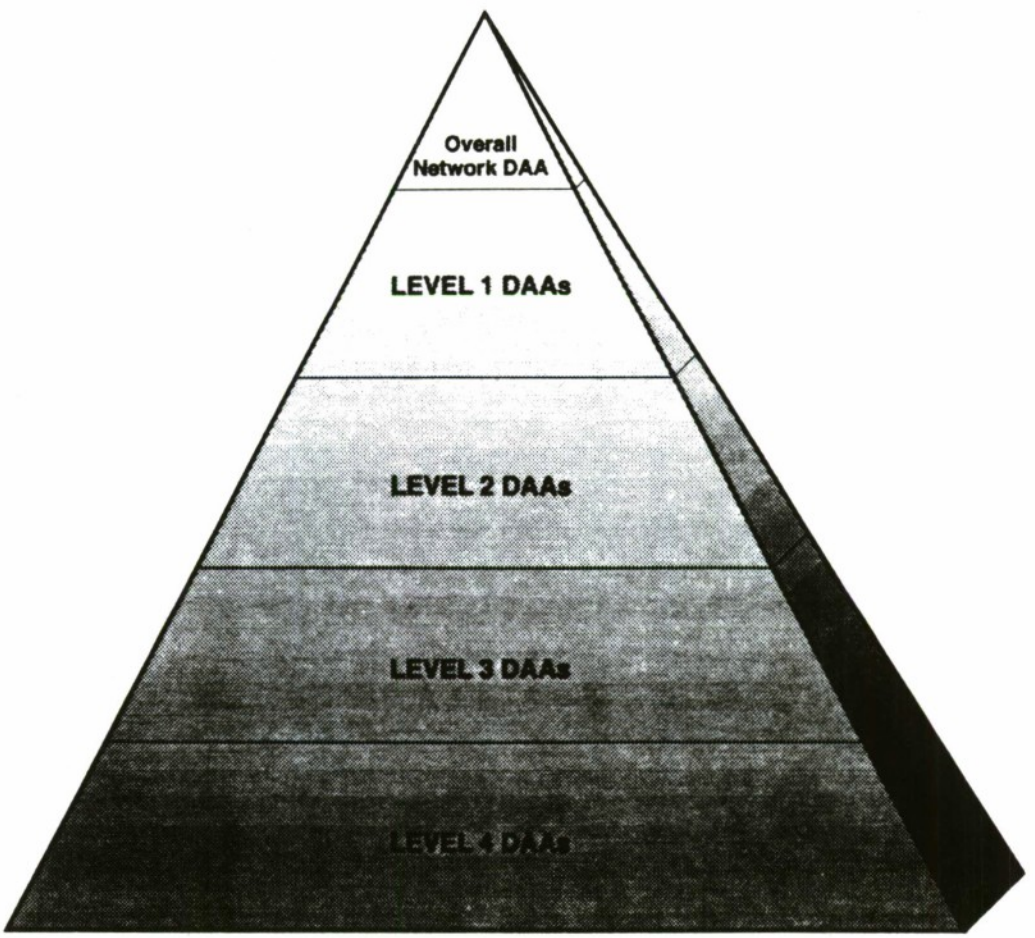


Figure 2. A "Pyramid" of DAAs

Step 2: Establish standard risk management procedures for security personnel to perform a risk analysis and provide certification to the DAA.

The risk analysis includes:

- The security environment assessment, which includes a criticality assessment to determine the relative importance of confidentiality, integrity, and availability; determining basic security facts such as data sensitivity and user clearances and restrictions; the required mode of operation; and the required TCB class.

- The risk assessment to analyze threats, vulnerabilities, and existing countermeasures to estimate the probabilities of threats exploiting vulnerabilities. The goal of risk assessment is to determine residual risks.

- The Security Test and Evaluation (ST&E), to verify that countermeasures are working properly to reduce threats and vulnerabilities to an acceptable level of risk.

- The countermeasure assessment to determine what additional countermeasures should be used (or, perhaps, are already planned to be used) to further minimize risks, and to determine the technical and economic feasibility of implementing the additional countermeasures.

Certification documentation includes all the written results of the risk analysis with a cover letter to the DAA requesting approval to operate. Some attachments (such as the TEMPEST countermeasure assessment) may be classified. This documentation:

- Certifies the ability of the system to meet the requirements of the Network Security Policy.

- Summarizes the result of the security environment assessment, including the criticality assessment, basic security facts, the security mode of operation, and the required TCB class.

- Identifies and quantifies residual risks which the DAA must accept before accrediting the system.

- Makes recommendations regarding the technical and economic feasibility of additional countermeasures which should be used (or are planned to be used) to further minimize risks to the system.

- Requests interim or final approval to operate.

Step 3: Establish procedures for component DAAs to grant approval to operate for their part of the network (or approval to connect for their affiliated systems), and provide certification to the next higher DAA.

Using Figure 1 as an example, key security personnel for each of the nine CPUs perform a risk analysis and provide certification to the nine DAAs. After each DAA accredits their individual systems, they provide certification to their respective network DAAs.

For each of the three networks, key security personnel examine the certification documentation provided by the DAAs for the standalone systems. A key security concern at this point must be the "cascading problem," which can result in serious security compromises. If cascading is a problem, they must consider countermeasures such as upgrading systems to a higher class, using guard processors, end-to-end encryption, etc. These key personnel perform a network risk analysis and provide certification to their respective network DAAs.

Each of the three network DAAs accredit their individual networks and provide certification to the overall network DAA. Security personnel who work for the overall network DAA examine the certification documentation provided by the three network DAAs. Then they perform a risk analysis for the entire "network of networks" and provide certification to the overall network DAA.

Obviously, this example based on Figure 1 is very simple; real-world networks are considerably more complex, and the process could involve many levels of iteration.

Step 4 achieves the final goal: the overall network DAA accredits the entire "network of networks," granting interim or final approval to operate. The approval documentation must specify what time period or what events will require reaccreditation of the entire network.

SUMMARY

Risk management of complex networks is difficult and resource intensive, but not an impossible task.

- During preliminary research, determine network boundaries, review existing documentation, and identify the overall network DAA.

- Publish a Network Security Policy (NSP) which covers the full spectrum of communications-computer system security disciplines and requirements.

- Construct a detailed security architecture which describes the mechanisms used to protect against threats and vulnerabilities. Identify security issues uncovered during this task.

- Develop an accreditation strategy for the network. Establish a security management structure to implement and enforce the NSP. Set up an iterative process for security personnel to perform risk analysis and provide certification documentation to the DAA. This process begins at the lowest levels of the security management structure and gradually encompasses more and more systems as successive DAAs accredit their portion of the network. Ultimately, the overall network DAA accredits the entire network.

REFERENCES

[1] DoD Directive 5200.28, Security Requirements for Automated Information Systems (AIS), 21 Mar 88.

[2] DoD 5200.28-STD, Department of Defense Trusted Computer System Evaluation Criteria ("Orange Book"), Dec 85.

[3] NCSC-TG-005, Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria ("Red Book"), 31 Jul 87.

[4] The National Research Council: Computers at Risk, 1991, page 140.

[5] Beer, Stafford, Management Science: The Business Use of Operations Research, 1968; Schoderbek, Peter et al., Management Systems: Conceptual Considerations, 1975.

[6] Salasin, John and Chilli, Frank, "Transition Engineering Methodology", IEEE Proceedings, 1990, page 110.

ACRONYMS

AIS	Automated Information System
BCSSM	Base Computer Systems Security Manager
COMSEC	Communications Security
CPU	Central Processing Unit
CSM	Computer Systems Manager
CSO	Communications-Computer Security Officer
CSSO	Computer Systems Security Officer
DAA	Designated Approving Authority
DBMS	Data Base Management System
DoD	Department of Defense
FAM	Functional Area Manager
IEEE	Institute of Electrical and Electronics Engineers
INFOSEC	Information Security
MAJCOM	Major Command
MCSSM	MAJCOM Computer Systems Security Manager
NCSC	National Computer Security Center
NM	Network Manager
NSO	Network Security Officer
NSP	Network Security Policy
OPSEC	Operations Security
ST&E	Security Test and Evaluation
TASO	Terminal Area Security Officer
TCB	Trusted Computing Base
TEMPEST	Emanations Security

Role-Based Access Controls

David Ferraiolo and Richard Kuhn

National Institute of Standards and Technology
Technology Administration
U.S. Department of Commerce
Gaithersburg, Md. 20899 USA

ABSTRACT

While Mandatory Access Controls (MAC) are appropriate for multi-level secure military applications, Discretionary Access Controls (DAC) are often perceived as meeting the security processing needs of industry and civilian government. This paper argues that reliance on DAC as the principal method of access control is unfounded and inappropriate for many commercial and civilian government organizations. The paper describes a type of non-discretionary access control - role-based access control (RBAC) - that is more central to the secure processing needs of non-military systems then DAC.

Keywords: access control, computer security, discretionary access control, integrity, mandatory access control, role, TCSEC

1. Introduction

The U.S. government has been involved in developing security technology for computer and communications systems for some time. Although advances have been great, it is generally perceived that the current state of security technology has, to some extent failed to address the needs of all. [1] , [2] This is especially true of organizations outside the Department of Defense (DoD). [3]

The current set of security criteria, criteria interpretations, and guidelines has grown out of research and development efforts on the part of the DoD over a period of twenty plus years. Today the best known U.S. computer security standard is the Trusted Computer System Evaluation Criteria (TCSEC [4]). It contains security features and assurances, exclusively derived, engineered and rationalized based on DoD security policy, created to meet one major security objective -- preventing the unauthorized observation of classified information. The result is a collection of security products that do not fully address security issues as they pertain to unclassified sensitive processing environments. Although existing security mechanisms have been partially successful in promoting security solutions outside of the DoD [2] , in many instances these controls are less then perfect, and are used in lieu of a more appropriate set of controls.

The TCSEC specifies two types of access controls: Discretionary Access Controls (DAC) and Mandatory Access Controls (MAC). Since the TCSEC's appearance in

December of 1983, DAC requirements have been perceived as being technically correct for commercial and civilian government security needs, as well as for single-level military systems. MAC is used for multi-level secure military systems, but its use in other applications is rare. The premise of this paper is that there exists a third type of access control, referred to as Role-Based Access Control (RBAC), that can be more appropriate and central to the secure processing needs within industry and civilian government than that of DAC, although the need for DAC will continue to exist.

2. Aspects of Security Policies

Recently, considerable attention has been paid to researching and addressing the security needs of commercial and civilian government organizations. It is apparent that significant and broad sweeping security requirements exist outside the Department of Defense. [2] , [5] , [6] Civilian government and corporations also rely heavily on information processing systems to meet their individual operational, financial, and information technology requirements. The integrity, availability, and confidentiality of key software systems, databases, and data networks are major concerns throughout all sectors. The corruption, unauthorized disclosure, or theft of corporate resources could disrupt an organization's operations and have immediate, serious financial, legal, human safety, personal privacy and public confidence impact.

Like DoD agencies, civilian government and commercial firms are very much concerned with protecting the confidentiality of information. This includes the protection of personnel data, marketing plans, product announcements, formulas, manufacturing and development techniques. But many of these organizations have even greater concern for integrity. [1]

Within industry and civilian government, integrity deals with broader issues of security than confidentiality. Integrity is particularly relevant to such applications as funds transfer, clinical medicine, environmental research, air traffic control, and avionics. The importance of integrity concerns in defense systems has also been studied in recent years. [7] , [8]

A wide gamut of security policies and needs exist within civilian government and private organizations. An organizational meaning of security cannot be presupposed. Each organization has unique security requirements, many of which are difficult to meet using traditional MAC and DAC controls.

As defined in the TCSEC and commonly implemented, DAC is an access control mechanism that permits system users to allow or disallow other users access to objects under their control:

> A means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control). [4]

DAC, as the name implies, permits the granting and revoking of access privileges to be left to the discretion of the individual users. A DAC mechanism allows users to grant or revoke access to any of the objects under their control without the intercession of a system administrator.

In many organizations, the end users do not "own" the information for which they are allowed access. For these organizations, the corporation or agency is the actual "owner" of system objects as well as the programs that process it. Control is often based on employee functions rather than data ownership.

Access control decisions are often determined by the roles individual users take on as part of an organization. This includes the specification of duties, responsibilities, and qualifications. For example, the roles an individual associated with a hospital can assume include doctor, nurse, clinician, and pharmacist. Roles in a bank include teller, loan officer, and accountant. Roles can also apply to military systems; for example, target analyst, situation analyst, and traffic analyst are common roles in tactical systems. A role based access control (RBAC) policy bases access control decisions on the functions a user is allowed to perform within an organization. The users cannot pass access permissions on to other users at their discretion. This is a fundamental difference between RBAC and DAC.

Security objectives often support a higher level organizational policy, such as maintaining and enforcing the ethics associated with a judge's chambers, or the laws and respect for privacy associated with the diagnosis of ailments, treatment of disease, and the administering of medicine with a hospital. To support such policies, a capability to centrally control and maintain access rights is required. The security administrator is responsible for enforcing policy and represents the organization.

The determination of membership and the allocation of transactions to a role is not so much in accordance with discretionary decisions on the part of a system administrator, but rather in compliance with organization-specific protection guidelines. These policies are derived from existing laws, ethics, regulations, or generally accepted practices. These policies are non-discretionary in the sense that they are unavoidably imposed on all users. For example, a doctor can be provided with the transaction to prescribe medicine, but does not possess the authority to pass that transaction on to a nurse.

RBAC is in fact a form of mandatory access control, but it is not based on multilevel security requirements. As defined in the TCSEC, MAC is

> A means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e. clearance) of subjects to access information of such sensitivity. [4]

Role based access control, in many applications (e.g. [9] , [10] , [11]) is concerned more with access to functions and information than strictly with access to information.

The act of granting membership and specifying transactions for a role is loosely analogous to the process of clearing users (granting membership) and the labeling (associate operational sensitivities) of objects within the DoD. The military policy is with respect to one type of capability: who can read what information. For these systems the unauthorized flow of information from a high level to a low level is the principal concern. As such, constraints on both reads and writes are in support of that rule. Within a role-based system, the principal concern is protecting the integrity of information: "who can perform what acts on what information."

A role can be thought of as a set of transactions that a user or set of users can perform within the context of an organization. Transactions are allocated to roles by a system administrator. Such transactions include the ability for a doctor to enter a diagnosis,

prescribe medication, and add a entry to (not simply modify) a record of treatments performed on a patient. The role of a pharmacist includes the transactions to dispense but not prescribe prescription drugs. Membership in a role is also granted and revoked by a system administrator.

Roles are group oriented. For each role, a set of transactions allocated the role is maintained. A transaction can be thought of as a transformation procedure [1] (a program or portion of a program) plus a set of associated data items. In addition, each role has an associated set of individual members. As a result, RBACs provide a means of naming and describing many-to-many relationships between individuals and rights. Figure 1 depicts the relationships between individual users, roles/groups, transformation procedures, and system objects.

The term transaction is used in this paper as a convenience to refer to a binding of transformation procedure and data storage access. This is not unlike conventional usage of the term in commercial systems. For example, a savings deposit transaction is a procedure that updates a savings database and transaction file. A transaction may also be quite general, e.g. "read savings file". Note however, that "read" is not a transaction in the sense used here, because the read is not bound to a particular data item, as "read savings file" is.

The importance of control over transactions, as opposed to simple read and write access, can be seen by considering typical banking transactions. Tellers may execute a savings deposit transaction, requiring read and write access to specific fields within a savings file and a transaction log file. An accounting supervisor may be able to execute correction transactions, requiring exactly the same read and write access to the same files as the teller. The difference is the process executed and the values written to the transaction log file.

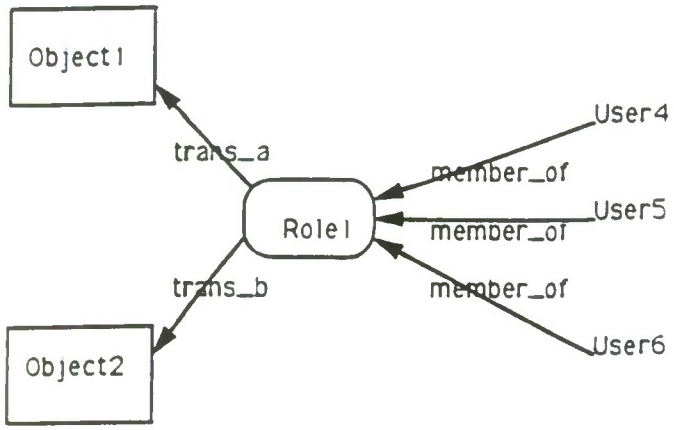


Figure 1 - Role relationships

The applicability of RBAC to commercial systems is apparent from its widespread use. Baldwin [9] describes a database system using roles to control access. Nash and

Poland [10] discuss the application of role based access control to cryptographic authentication devices commonly used in the banking industry. Working with industry groups, the National Institute of Standards and Technology has developed a proposed standard, "Security Requirements for Cryptographic Modules," (Federal Information Processing Standard 140-1) [11] that will require support for access control and administration through roles. To date, these role based systems have been developed by a variety of organizations, with no commonly agreed upon definition or recognition in formal standards. *Role based access controls described in this paper address security primarily for application-level systems, as opposed to general purpose operating systems.*

3. Formal Description of RBAC

To clarify the notions presented in the previous section, we give a simple formal description, in terms of sets and relations, of role based access control. No particular implementation mechanism is implied.

For each subject, the active role is the one that the subject is currently using:

$AR(s{:}subject)$ = {the active role for subject s}.

Each subject may be authorized to perform one or more roles:

$RA(s{:}subject)$ = {authorized roles for subject s}.

Each role may be authorized to perform one or more transactions:

$TA(\{r{:}role\})$ = {transactions authorized for role r}.

Subjects may execute transactions. The predicate $exec(s,t)$ is true if subject s can execute transaction t at the current time, otherwise it is false:

$exec(s{:}subject, t{:}tran)$ = true iff subject s can execute transaction t.

Three basic rules are required:

(1) Role assignment: A subject can execute a transaction only if the subject has selected or been assigned a role:

$\forall s{:}subject, t{:}tran\,(exec(s,t) \Rightarrow AR(s) \neq \varnothing)$.

The identification and authentication process (e.g. login) is not considered a transaction. All other user activities on the system are conducted through transactions. Thus all active users are required to have some active role.

(2) Role authorization: A subject's active role must be authorized for the subject:

$\forall s{:}subject\,(AR(s) \subseteq RA(s))$.

With (1) above, this rule ensures that users can take on only roles for which they are authorized.

(3) Transaction authorization: A subject can execute a transaction only if the transaction is authorized for the subject's active role:

$\forall s{:}subject, t{:}tran\,(exec(s,t) \Rightarrow t \in TA(AR(s)))$.

With (1) and (2), this rule ensures that users can execute only transactions for which they are authorized. Note that, because the conditional is "only if", this rule allows the possibility that additional restrictions may be placed on transaction execution. That is, the rule does not guarantee a transaction to be executable just because it is in $TA(AR(s))$, the set of

transactions potentially executable by the subject's active role. For example, a trainee for a supervisory role may be assigned the role of "Supervisor", but have restrictions applied to his or her user role that limit accessible transactions to a subset of those normally allowed for the Supervisor role.

In the preceding discussion, a transaction has been defined as a transformation procedure, plus a set of data items accessed by the transformation procedure. Access control in the rules above does not require any checks on the user's right to access a data object, or on the transformation procedure's right to access a data item, since the data accesses are built into the transaction. Security issues are addressed by binding operations and data into a transaction at design time, such as when privacy issues are addressed in an insurance query transaction.

It is also possible to redefine the meaning of "transaction" in the above rules to refer only to the transformation procedure, without including a binding to objects. This would require a fourth rule to enforce control over the modes in which users can access objects through transaction programs. For example, a fourth rule such as

(4) $\forall s:subject, t:tran, o:object\,(exec(s,t) \Rightarrow access(AR(s),t,o,x))$

could be defined using a transaction (redefined to transformation procedure) to object access function $access(r,t,o,x)$ which indicates if it is permissible for a subject in role r to access object o in mode x using transaction t, where x is taken from some set of modes such as read, write, append. Note that the Clark-Wilson access control triple could be implemented by letting the modes x be the access modes required by transaction t, and having a one-to-one relationship between subjects and roles. RBAC, as presented in this paper, thus includes Clark and Wilson access control as a special case.

Use of this fourth rule might be appropriate, for example, in a hospital setting. A doctor could be provided with read/write access to a prescription file, while the hospital pharmacist might have only read access. (Recall that use of the first three rules alone requires binding the transaction program t and data objects that t can access, and only controls access to the transactions.) This alternative approach using the fourth rule might be helpful in enforcing confidentiality requirements.

Another use of RBAC is to support integrity. Integrity has been defined in a variety of ways, but one aspect [8] of integrity is a requirement that data and processes be modified only in authorized ways by authorized users. This seems to be a reasonable security objective for many real systems, and RBAC should be applicable to such systems.

In general, the problem of determining whether data have been modified only in authorized ways can be as complex as the transaction that did the modification For this reason, the practical approach is for transactions to be certified and trusted. If transactions must be trusted then access control can be incorporated directly into each transaction. Requiring the system to control access of transaction programs to objects through the access function used in rule (4) might then be a useful form of redundancy, but it could involve significant overhead for a limited benefit in enforcing integrity requirements. Therefore, inclusion of a transaction to object access control function in RBAC would be useful in some, but not all applications.

4. Centrally Administering Security Using RBAC

RBAC is flexible in that it can take on organizational characteristics in terms of policy and structure. One of RBAC's greatest virtues is the administrative capabilities it supports.

Once the transactions of a Role are established within a system, these transactions tend to remain relatively constant or change slowly over time. The administrative task consists of granting and revoking membership to the set of specified named roles within the system. When a new person enters the organization, the administrator simply grants membership to an existing role. When a person's function changes within the organization, the user membership to his existing roles can be easily deleted and new ones granted. Finally, when a person leaves the organization, all memberships to all Roles are deleted. For an organization that experiences a large turnover of personnel, a role-based security policy is the only logical choice.

In addition, roles can be composed of roles. For example, a Healer within a hospital can be composed of the roles Healer, Intern, and Doctor. Figure 2 depicts an example of such a relationship.

By granting membership to the Role Doctor, it implies access to all transactions defined by Intern and Healer, as well as those of a Doctor. On the other hand, by granting membership to the Intern role, this implies transactions of the Intern and Healer not the Doctor. However, by granting membership to the Healer role, this only allows access to those resources allowed under the role Healer.

5. Principle of Least Privilege

The principle of least privilege has been described as important for meeting integrity objectives. [8] The principle of least privilege requires that a user be given no more privilege than necessary to perform a job. Ensuring least privilege requires identifying what the user's job is, determining the minimum set of privileges required to perform that job, and restricting the user to a domain with those privileges and nothing more. By denying to subjects transactions that are not necessary for the performance of their duties, those denied privileges cannot be used to circumvent the organizational security policy. Although the concept of least privilege currently exists within the context of the TCSEC, requirements restrict those privileges of the system administrator. Through the use of RBAC, enforced minimum privileges for general system users can be easily achieved.

6. Separation of Duties

RBAC mechanisms can be used by a system administrator in enforcing a policy of separation of duties. Separation of duties is considered valuable in deterring fraud since fraud can occur if an opportunity exists for collaboration between various job related capabilities. Separation of duty requires that for particular sets of transactions, no single individual be allowed to execute all transactions within the set. The most commonly used examples are the separate transactions needed to initiate a payment and to authorize a payment. No single individual should be capable of executing both transactions. Separation of duty is an important consideration in real systems. [1] , [12] , [13] , [14] The sets in question will vary depending on the application. In real situations, only

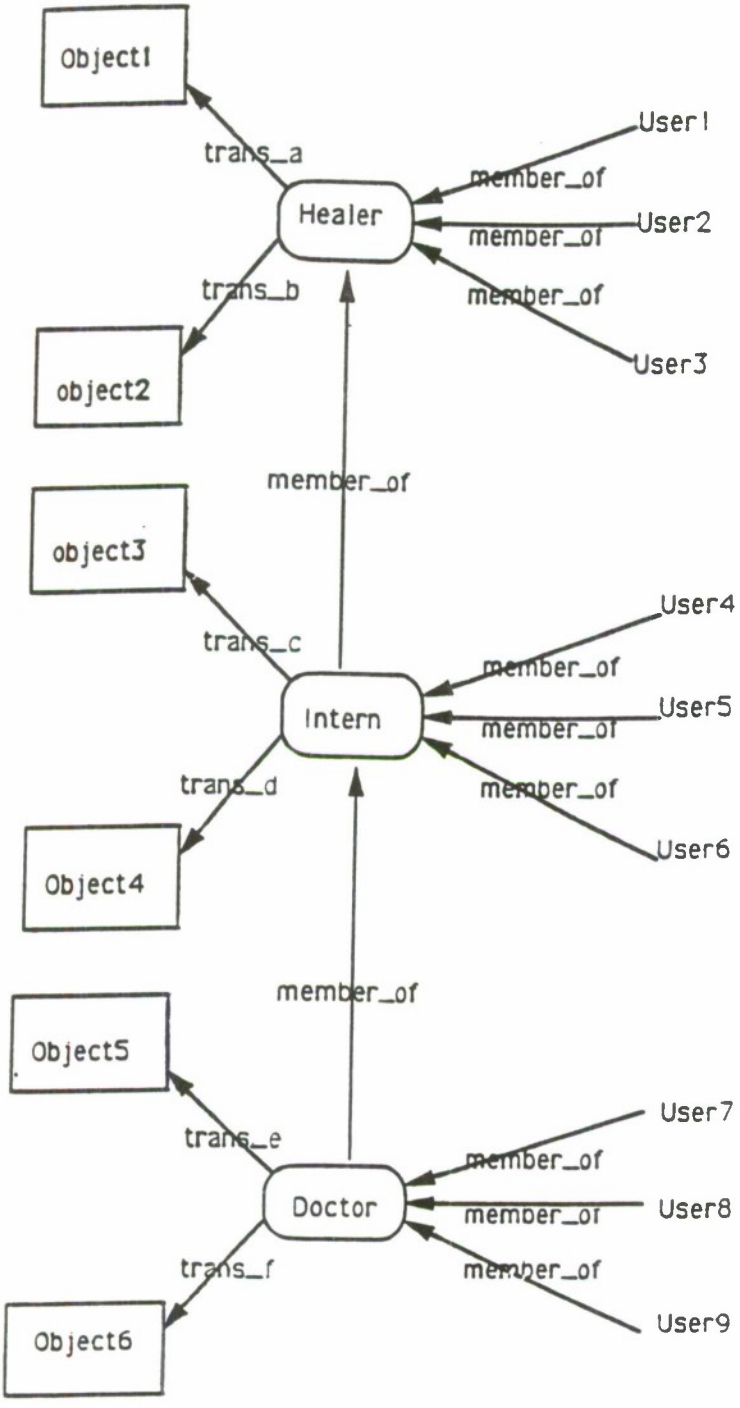


Figure 2 - Multi-Role relationships

certain transactions need to be restricted under separation of duty requirements. For example, we would expect a transaction for "authorize payment" to be restricted, but a transaction "submit suggestion to administrator" would not be.

Separation of duty can be either static or dynamic. Compliance with static separation requirements can be determined simply by the assignment of individuals to roles and allocation of transactions to roles. The more difficult case is dynamic separation of duty.

where compliance with requirements can only be determined during system operation. The objective behind dynamic separation of duty is to allow more flexibility in operations. Consider the case of initiating and authorizing payments. A static policy could require that no individual who can serve as payment initiator could also serve as payment authorizor. This could be implemented by ensuring that no one who can perform the initiator role could also perform the authorizer role. Such a policy may be too rigid for commercial use, making the cost of security greater than the loss that might be expected without the security. More flexibility could be allowed by a dynamic policy that allows the same individual to take on both initiator and authorizer roles, with the exception that no one could authorize payments that he or she had initiated. The static policy could be implemented by checking only roles of users; for the dynamic case, the system must use both role and user ID in checking access to transactions.

Separation of duty is necessarily determined by conditions external to the computer system. The Clark-Wilson [1] scheme includes the requirement that the system maintain the separation of duty requirement expressed in the access control triples. Enforcement is on a per-user basis, using the user ID from the access control triple. As discussed above, user functions can be conveniently separated by role, since many users in an organization typically perform the same function and have the same access rights on TPs and data. Allocating access rights according to role is also helpful in defining separation of duty in a way that can be enforced by the system.

7. Summary and Conclusions

In many organizations in industry and civilian government, the end users do not "own" the information for which they are allowed access. For these organizations, the corporation or agency is the actual "owner" of system objects, and discretionary access control may not be appropriate. Role-Based Access Control (RBAC) is a non-discretionary access control mechanism which allows and promotes the central administration of an organizational specific security policy.

Access control decisions are often based on the roles individual users take on as part of an organization. A role specifies a set of transactions that a user or set of users can perform within the context of an organization. RBAC provide a means of naming and describing relationships between individuals and rights, providing a method of meeting the secure processing needs of many commercial and civilian government organizations.

Various forms of role based access control have been described and some are used in commercial systems today, but there is no commonly accepted definition or formal standards encompassing RBAC. As such, evaluation and testing programs for these systems have not been established as they have for systems conforming to the Trusted Computer Security Evaluation Criteria. This paper proposed a definition of The requirements and access control rules for RBAC proposed in this paper could be used as the basis for a common definition of access controls based on user roles.

References

1. D.D. Clark and D.R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," *IEEE Symposium on Computer Security and Privacy*, April, 1987.

2. National Research Council, *Computers at Risk,* National Academy Press, 1991.

3. National Institute of Standards and Technology, *Minimum Security Functionality Requirements for Multi-User Operating Systems (draft),* Computer Systems Laboratory, NIST, January 27, 1992.

4. Department of Defense, *Trusted Computer Security Evaluation Criteria, DOD 5200.28-STD*, 1985.

5. Z.G. Ruthberg and W.T. Polk, Editors, *Report of the Invitational Workshop on Data Integrity,* Natl. Inst. of Stds. and Technology, SP 500-168, 1989.

6. S.W. Katzke and Z.G. Ruthberg, Editors, *Report of the Invitational Workshop on Integrity Policy in Computer Information Systems,* Natl. Inst. of Stds. and Technology, SP 500-160, 1987.

7. J.E. Roskos, S.R. Welke, J.M. Boone, and T. Mayfield, *Integrity in Tactical and Embedded Systems,* Institute for Defense Analyses, HQ 89-034883/1, October, 1989.

8. National Computer Security Center, *Integrity in Automated Information Systems* , September, 1991.

9. R.W. Baldwin, "Naming and Grouping Privileges to Simplify Security Management in Large Databases," *IEEE Symposium on Computer Security and Privacy*, 1990.

10. M.J. Nash and K.R. Poland, "Some Conundrums Concerning Separation of Duty," *IEEE Symposium on Computer Security and Privacy*, 1990.

11. National Institute of Standards and Technology, *Security Requirements for Cryptographic Modules,* Natl. Inst. of Stds. and Technology, FIPS 140-1, 1992.

12. W.R. Shockley, "Implementing the Clark/Wilson Integrity Policy Using Current Technology," *Proceedings of 11th National Computer Security Conference*, October, 1988.

13. R. Sandhu, "Transaction Control Expressions for Separation of Duties," *Fourth Aerospace Computer Security Applications Conference*, December, 1988.

14. P. Terry and S. Wiseman, "A 'New' Security Policy Model," *IEEE Symposium on Computer Security and Privacy*, May, 1989.

AN SDNS PLATFORM FOR TRUSTED PRODUCTS

Ernie Borgoyne
Motorola Inc., Government Electronics Group
8201 E. McDowell Road, Mail Stop H2250
Scottsdale, Arizona 85252

Ralph G. Puga
Trusted Information Systems, Inc.
3060 Washington Road (Rt. 97)
Glenwood, Maryland 21738

ABSTRACT

The Network Encryption System (NES) security platform is designed with an open architecture that allows commercially available trusted products to be easily integrated with Secure Data Network System (SDNS) technology. SDNS is the U.S. government's new mainstream secure networking technology that offers significant cost benefits to users because of its scalability and interoperability of services. This paper presents a brief overview of the Secure Data Network System and a description of the SDNS based Network Encryption System. It then discusses the work that is being done in developing a trusted SDNS interface for the NES by integrating Trusted Xenix™ onto the NES security platform.

INTRODUCTION

The NES was evaluated by the National Security Agency (NSA) under the Commercial COMSEC Endorsement Program (CCEP). The NES is the first product designed to SDNS standards, endorsed by NSA, and available today. The open architecture design of the NES security platform provides the ability to integrate standard commercially available networking and security technology, including network media interface boards, network routing and management software, and COMPUSEC evaluated trusted computer products. The benefit of the NES open COMSEC architecture, is that it provides an SDNS platform that can be used as a foundation for a variety of new INFOSEC applications and products. Figure 1 illustrates the essence of this commercial technology utilization by the NES security platform.

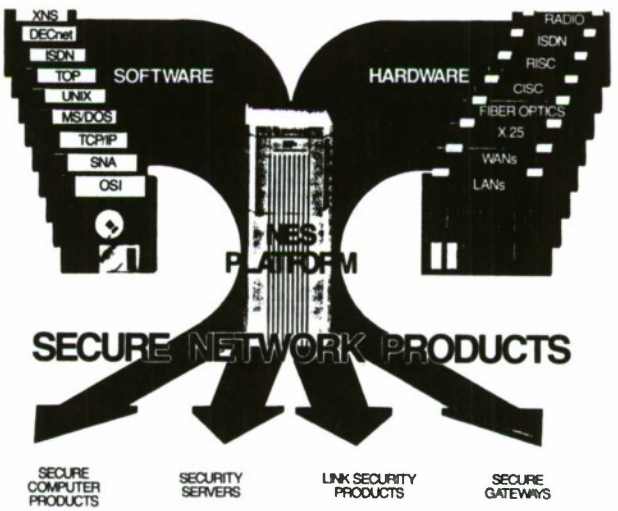


Figure 1 INFOSEC PRODUCT Platform

This paper will first present an overview of SDNS for the benefit of those who are not familiar with this new technology. The NES will then be presented including a description of the NES security platform's basic open architecture, which will allow the reader to understand how the new SDNS technology can be easily utilized. A new configuration will then be presented called the INFOSEC Computer Platform. The INFOSEC Computer Platform incorporates an embedded 386-based processor board within the NES security platform, running Trusted Xenix™ and a trusted interface to the SDNS functions. This interface will allow user application software together with a B2+ level secure operating system to be integrated within the same physical environment. As the product evolves, this SDNS interface can be extended to allow secure end-to-end communications via standard networks through the use of networking applications. This paper will conclude by presenting some potential applications for this proposed INFOSEC platform.

SECURE DATA NETWORK SYSTEM

SDNS is a set of standards for interoperability of data security devices over public and private data networks. The SDNS standards were developed by a U.S. government and industry consortium, and sponsored by the National Security Agency. Established in 1986, the consortium has produced a set of security protocol specifications for the application of security functions at various layers in the network communication stack. The SDNS protocols are based on the International Standards Organization (ISO) - Open Systems Interconnection (OSI) reference model, and provide security services designed to protect both government Classified (Type I) and government Unclassified-Sensitive (Type II) information while being transmitted over Unclassified networks.

The heart of the SDNS operation is the NSA's Electronic Key Management System (EKMS), and is illustrated in Figure 2. The EKMS is a national resource that supports U.S. government users and its contractors, in the enforcement of their security policies for handling Type I and II information between their automated data processing systems. Users order key material for each SDNS device within their organization. In response, the EKMS provides a non-forgeable certificate plus keying material, which are then loaded into the SDNS device by the user during initialization. The certificates contain the user-specified rule-based access control policy for that device to enforce.

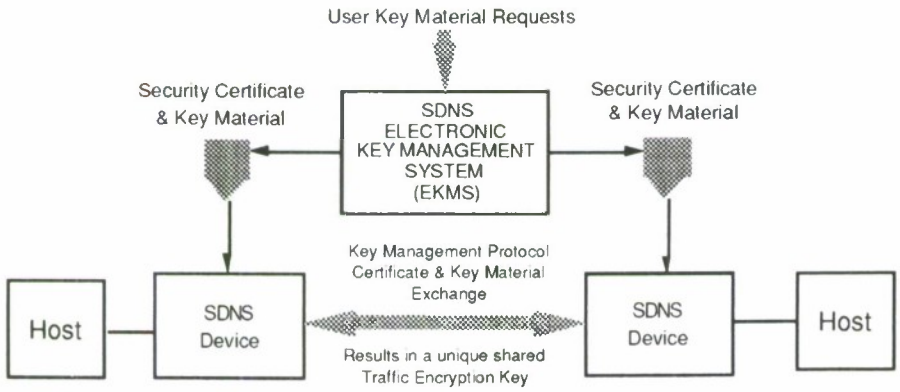


Figure 2 SDNS Key Management and Distribution

The SDNS Key Management Protocol (KMP) provides a powerful authentication and access control mechanism based on the exchange of the non-forgeable certificates between SDNS security devices. KMP runs at the application layer on top of a full seven-layer OSI stack within

each SDNS device. During the certificate exchange process, KMP compares the fields to determine if there is an intersection of security attributes. If there is an intersection, the keying material is used by the SDNS key generation algorithms to create a unique pair-wise Traffic Encryption Key (TEK) that is known only by those two devices. The TEK represents a secure transmission channel between two SDNS devices that can be used for data that meets the security criteria established during Key Management Protocol exchange.

Data is transmitted over the secure transmission channel using a security protocol designed for operation at a particular layer in the communication stack. At the transport and network layers, SDNS has defined Security Protocol layer 4 (SP4) and Security Protocol layer 3 (SP3) respectively. These security protocols provide confidentiality by encryption using the established TEK, integrity, and access control services for data at their respective layers. Proper handling of labelled data by the security protocol is required to ensure that information is sent over the correct channel.

In summary, SDNS provides a powerful set of tools for enforcing organizational security policies dealing with the handling of data within automated processing systems serving all of government and its contractors. The possiblities for interoperability among users is greatly increased by SDNS, first because it is based on standards, and second because the Electronic Key Management System (EKMS) is a national resource serving all of government and industry. The EKMS is scalable because, unlike earlier key management systems which are required to know about all authorized pair-wise connections, it is based on a rule-based access control system.

The STU-III is an excellent example of the significance of interoperability and scalability to information security. Approximately 250,000 STU-III terminals are deployed world-wide, and virtually any pair of users with the proper clearance can communicate. SDNS is revolutionizing data security, just as the STU-III has revolutionized voice security.

NETWORK ENCRYPTION SYSTEM

The Network Encryption System (NES) is a family of components offering a set of tools for enforcing the user's automated data processing system security policy. The major components include the SDNS Electronic Key Management System (EKMS), the NES Product Server, and the NES security server. These components are shown in the Product Family in Figure 3.

The EKMS provides SDNS keying material to the NES security servers either in the form of a physical operational key or seed key. The initial keying material is contained on a KSD-64A (physically identical to a STU-III key) and loaded during device initialization. A Seed key is used to establish a secure connection with the EKMS to receive operational key electronically. The operational keying material, which is valid for one year, includes the non-forgeable certificate with identification and security attributes, and information for generating Traffic Encryption Keys (TEKs) between pairs of NES security servers.

The NES Product Server provides Administration, Discretionary Access Control, and Audit functions to support a domain of NES security servers. The system administrator enters the configuration for each device, including addressing and access control information, then generates a configuration disk. This disk containing application software, static routing tables and identity based access control tables, is loaded into the security server during initialization.

An NES security server provides SDNS services to one or more Hosts/Workstations running on a RED side LAN, and connects to a LAN or WAN on the unclassified BLACK side. The security server application of the NES will be discussed in more detail, following a discussion of the basic NES security platform architecture.

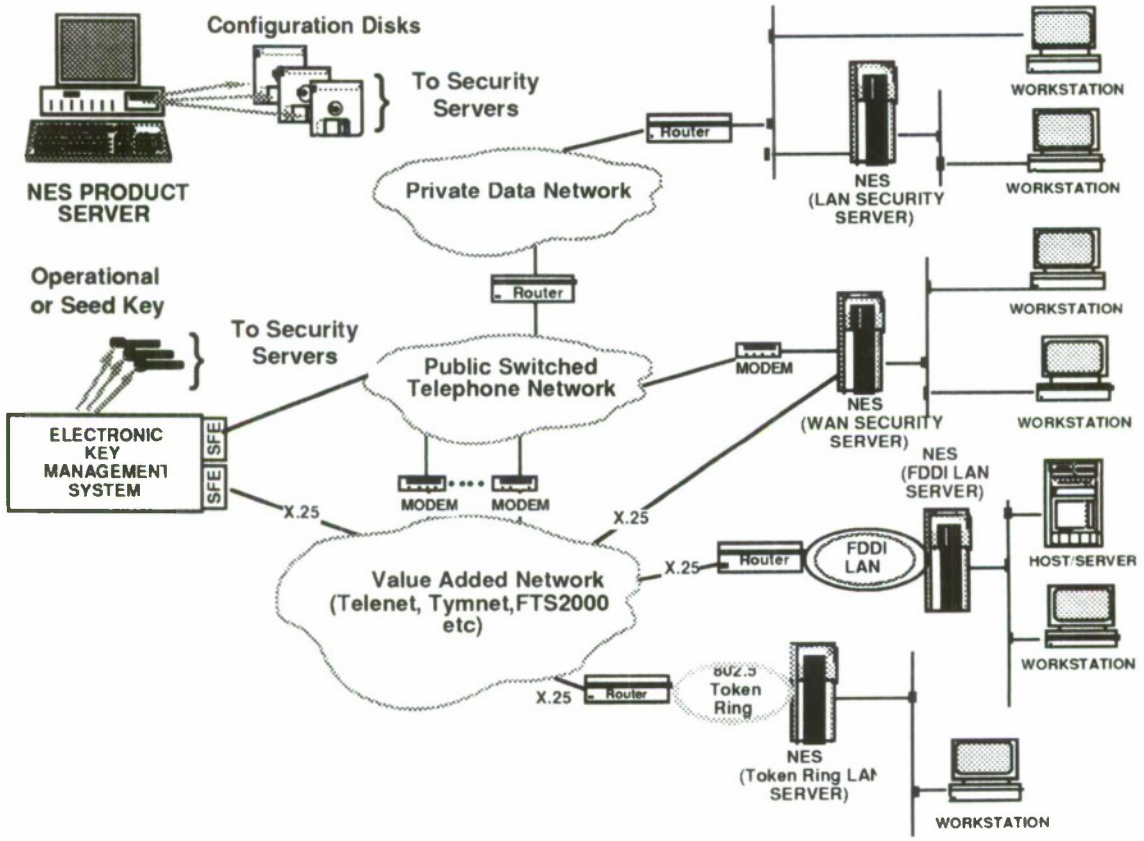


Figure 3 NES Product Family

NES Security Platform Architecture

The NES security platform architecture shown in Figure 4, is a self-contained Tamper and TEMPEST protected device with a front panel, and host, network and diagnostic interfaces on the back panel. The front panel provides a keyceptacle for loading key material from the KSD-64A data key supplied by the EKMS. Also, there is a disk drive for loading the configuration information and application software contained on the configuration disk generated at the NES Product Server. The key material and configuration disk are read by the security platform during the start-up process. A cryptographic checksum, based on the key material and the configuration information, is generated and written back to the disk binding the disk to the security platform.

Internally, the NES security platform contains a security kernel, and a separate RED and BLACK VMEbus[1] that can accomodate standard commercially available VMEbus boards for running the application software. This concept is shown also in Figure 4. Four VME boards can be supported on the RED side, and two on the BLACK. The security kernel provides the basic SDNS functions to support different applications and enforces the required COMSEC assurances. Common Environment (CE) is the internal operating system running both on the VME I/O boards and in the kernel allowing task-to-task and board-to-board communication to occur, and allowing tasks running on the processor boards to invoke the SDNS services that are provided by the security kernel.

[1] VME (Versa Module Europe) is an IEEE standard defined by the P1014, IEC 47b working committee.

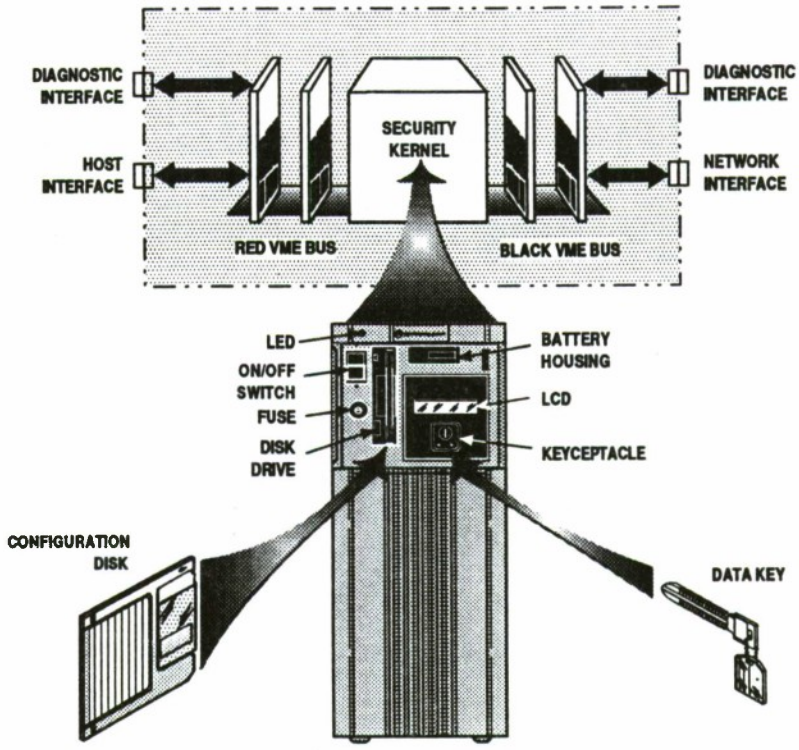


Figure 4 NES Security Platform Architecture

NES Security Server Applications

In the current applications, the NES security platform is viewed as a single-level secure telecommunications server for hosts or workstations residing on a classified Local Area Network (LAN). The RED side of the security server supports a LAN media interface allowing classified hosts residing on the LAN to send datagrams to the security server for processing. The security server then performs a source/destination address verification, adds an integrity checksum, encrypts the data, and sends the resulting unclassified data to the BLACK side. The BLACK side of the security server supports a LAN, or other telecommunication media interface, and adds a network layer header allowing the encrypted data to be sent to another remote SDNS device over an unclassified network. The unclassified network can be a single network or multiple networks (e.g., Internet), and can be made up of one or more LANs and/or Wide Area Networks (WANs). The ability to pass classified information over existing unclassified telelecommunication systems can offer a tremendous cost savings advantage to users.

The initial product (endorsed on March 13, 1991) supports an 802.3 (Ethernet) interface on both the RED and BLACK sides of the NES security server. The upper right portion of the NES Product Family in Figure 3 shows the Ethernet LAN server in a network environment. Three versions of network application software are also provided for this server including DoD IP, OSI, and Transparent (for supporting proprietary network protocol environments). The Wide Area Network (WAN) security server product, also shown in Figure 3, provides an Ethernet interface on the RED side and an X.25 interface on the BLACK side. The WAN server application allows direct X.25 connection to a Wide Area Network eliminating the need for an external router. The ability to eliminate expensive network devices with the NES security device offers significant cost advantages and drives down the cost of security. Two versions of application software are currently supported with this product including DoD IP and OSI. Other interfaces planned include FDDI and 802.5 (Token Ring).

INFOSEC COMPUTER PLATFORM

The INFOSEC Computer Platform utilizes the basic NES security platform architecture described above. However, unlike the NES security server applications previously described, which provide a host-side LAN media interface on the RED side, the INFOSEC Platform replaces the media interface board on the RED side with a computer processor board capable of supporting a variety of external peripherals and an internal hard disk. The INFOSEC Computer Platform Block Diagram including the security boundaries is shown in Figure 5.

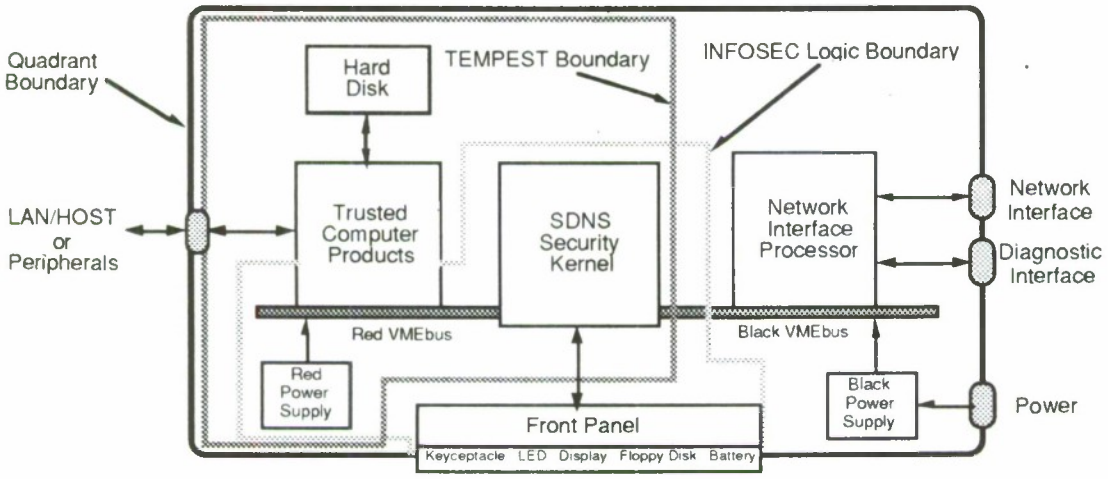


Figure 5 INFOSEC Computer Platform Block Diagram

SDNS Security Kernel Interface and Services

The security kernel interfaces electrically with the RED and BLACK VMEbus using the standard VMEbus protocol, and functionally using Common Environment (CE). Application software uses CE to invoke the following security kernel services:

Key Management Services These services are used to create and manage Traffic Encryption Keys (TEKs) that are created by the SDNS Key Management Protocol during the process of exchanging key material credentials with other SDNS devices. The TEKs are identified by a Key ID and provide a confidentiality service to applications running on the RED side at different security levels. A RED application may request that a TEK be created for a single security level or for a range of security levels, and for one or more compartments.

System Management Services These services are used during initialization and operation to configure and manage information on the RED and BLACK processor boards.

Application Control and Presentation Layer Services These services are used to manage connections initiated on the RED side between NES devices on the BLACK side. The NES identification, which is known on the RED side, is sent to the security kernel. This identification is then sent to the BLACK side where an Association is established with the destination NES using the seven layer OSI communication stack and the statically loaded network addressing information. This association is then used by the SDNS Key Management Protocol during key creation.

Encrypt/Decrypt Services These services allow security protocols running on the RED and BLACK sides to encrypt or decrypt traffic using an established TEK identified by Key ID, which provide integrity. The TEK represents a cryptographic channel for passing single or multi-level information over the Unclassified BLACK side.

Logical Task Flow Description

The INFOSEC Computer Platform logical flow is shown in Figure 6. The RED processor is a VMEbus compatible 386-based processor board running the Trusted Xenix™ Operating System, the Trusted SDNS interface software, and application software. The operating system, SDNS interface software and application software can be loaded from the internal disk (default), or the external floppy disk. The Trusted SDNS interface provides a trusted Common Environment interface between the security kernel and Trusted Xenix™ allowing applications at different security levels running on the RED side to invoke the security kernel services.

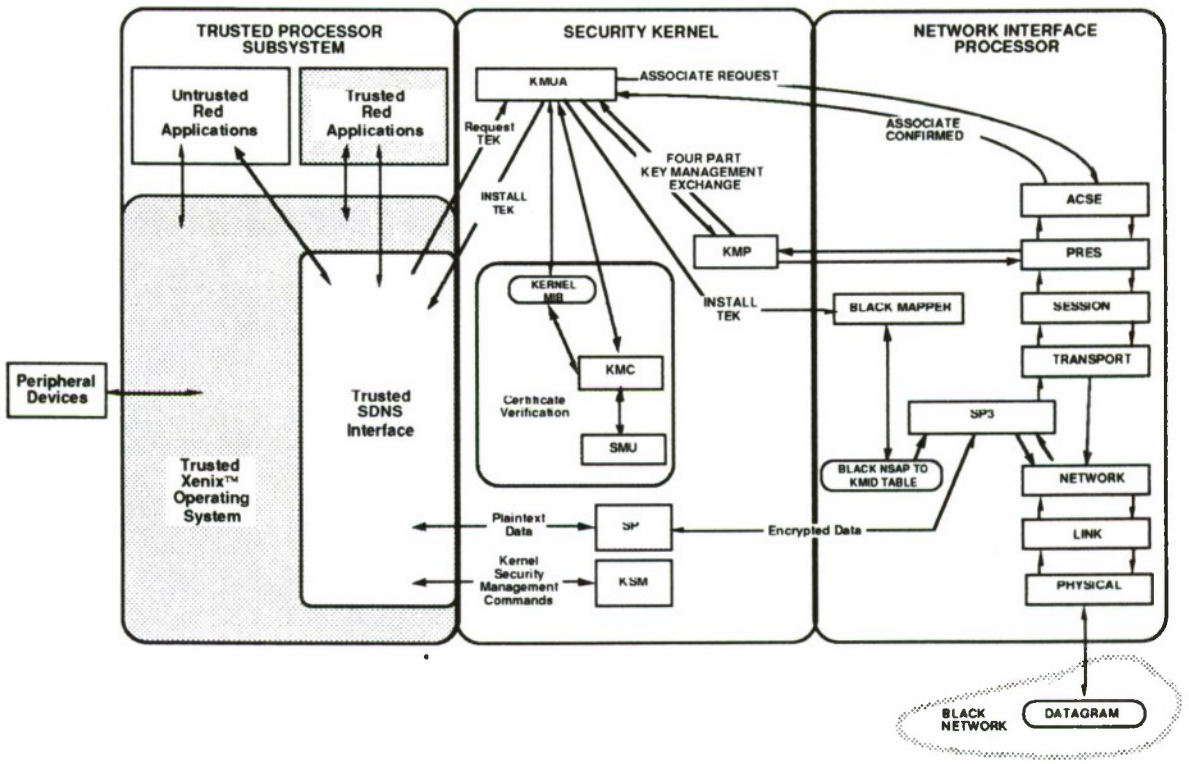


Figure 6 Logical Flow Diagram

The Security kernel runs a firmware implementation of the SDNS functions including the Key Management User Association (KMUA) process for Key Management, Application Control and Presentation services; Key Management Protocol (KMP) for performing the SDNS Identification & Authentication and key create functions; and the Security Protocol (SP) for performing the basic encryption/decryption functionality.

The Black Interface processor is one of a set of VMEbus compatible processor boards capable of providing a media interface as described earlier. The processor runs application software loaded from the security platform floppy disk during the INFOSEC Computer start-up process.

APPLICATIONS and CAPABILITIES

The INFOSEC Computer Platform is an evolutionary product built on previous CCEP and TPEP technology. This section discusses some potential applications and capabilities for this proposed INFOSEC platform, and suggests an evolutionary path to total communication security. Based on available hardware that can currently be integrated onto the INFOSEC Platform, three configurations are currently under consideration for development and NSA evaluation: the Multi-Level Secure (MLS) Workstation, the Secure Gateway, and the Communication Port Server.

Multi-Level Secure (MLS) Workstation Application

The MLS Workstation includes a monitor, keyboard, floppy disk, and optionally a printer, mouse, and streaming tape drive. The Trusted Xenix™ Operating System and user applications are loaded from an internal hard disk. An application currently under development is the Enhanced Product Server (EPS).

The EPS, in addition to performing all of the basic NES Product Server functions, will allow a System Administrator to remotely manage Identity Based Access Control (IBAC) and Audit information, and perform other system administration functions with NES Security Servers. This application is illustrated in Figure 7. EPS management software runs on the Trusted Operating system on the RED side of the MLS Workstation and communicates with EPS agents running in application software on the NES security servers located througout the network. The EPS agents collect audit information and respond to commands from the EPS manager to send audit, receive IBAC table updates, and perform other system functions.

The EPS and NES security servers all communicate over a common unclassified network. The NES security servers provide SDNS security services to LANs or hosts running at a single security level. Each NES security server allows traffic to pass between it and another device keyed at the same security level. The EPS will communicate with all NES security servers keyed at different security levels. The EPS allows a user logged in at the Top-Secret level to manage the data-base on the Red side of the Top-Secret NES security platform, and a user logged in at the Secret level to manage the data-base on the Red side of the Secret NES security platform.

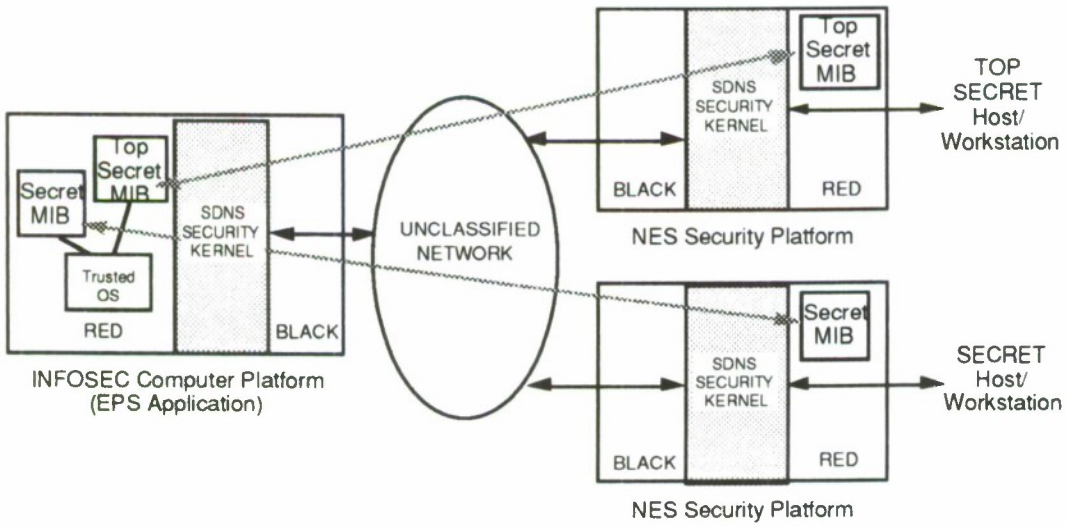


Figure 7 Enhanced Product Server Application

An NES application which can use the EPS capabilities is the Defense Simulation Internet (DSI). The DSI is a worldwide network which consists of a set of subnets operating at different security levels within the network domain. Each subnet is run by a local site administrator who receives direction from a central system administrator. Currently there are two approaches for administering this system.

The first approach involves putting an NES Product Server at each of the subnets to allow the site administrator to locally create configuration disks for his set of local NES Security Servers. Because each NES device requires a unique configuration diskette, which specifies all other NES devices with which it is allowed to communicate - both within the local subnet and worldwide, it is necessary for the site administrator of each NES grouping to duplicate the environment to resemble all of the NES devices within a particular grouping. The second approach involves using a single NES Product Server located at the system administrator's central site, to generate the configuration disks for all NES devices and distribute them to each subnet. This approach can be very time consuming due to mail delays, diskette failures, or as a result of improper mail delivery. Since this scenario involves a worldwide network the central NES administrator option seems impractical.

As one can see, each approach has its own drawbacks. In the first case, each site must contain an NES Product Server, which can mean a more costly configuration. In the second case, mail delays and other obstacles related to NES network configuration updates can result in lost time and or money. In either case there is a lot of room for error or loss of time and or money. A situation where errors could be propagated would be due to incorrect transmission of NES network grouping updates from the central NES Product Server or by mail/email.

The application described above could be resolved much more efficiently and more cost effectively with the EPS. If there were an EPS being used in this situation a Product Server at each site would no longer be necessary. Additionally, with the EPS' multilevel capability, only one EPS would be required for both of the subnets which were operating at different security levels. Finally, the risk range for a B2 level of operation would be satisfied.

Secure Gateway Capabilities

The Secure Gateway configuration provides multiple Local Area Network interfaces allowing local area networks running in a dedicated or system-high mode to communicate securely with other remote networks or systems. The Trusted Xenix™ operating system and applications are loaded from an internal hard disk. An application currently being developed for the Secure Gateway configuration is a Trusted Guard. This application will allow two classified networks to operate over an unclassified network and is illustrated in Figure 8.

The basic guard function consists of the Trusted Xenix™ Operating system which controls the access to a Secret Local Area Network or Host, and a Top-Secret Local Area Network or Host. A regrader function, executing as a trusted application, reads data from the higher Top Secret network and analyzes the information for compliance with a rule set. If the information succeeds the rule set analysis, the application performs a trusted write-down, allowing information to flow to the Secret side. By integrating this function into the INFOSEC Computer Platform, the function can be extended over an unclassified network.

Depending on the particular situation, the trusted write-down can be either performed manually or automatically. The manual review before write-down can be implemented by a trusted reviewer (e.g., using a UNIX-like 'more' utility via the trusted path) and can be implemented to reject or accept messages based on content. An automatic trusted write-down can be performed

with a process which scans messages based on strict formatting message content. In either case Trusted Xenix auditing can be performed in order to log all security relevant events.

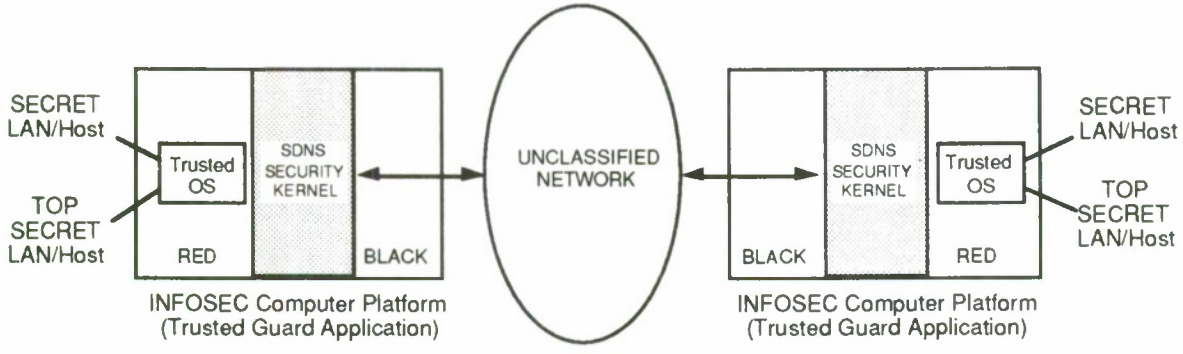


Figure 8 Trusted Guard Application

Communication Port Server Configuration

The Communication Port Server configuration provides multiple RS-232 communication ports allowing Terminals, Hosts, or Workstations, authorized for different security levels, or compartments, to communicate via the SDNS functions. It is envisioned that various multilevel and single level operations can be supported using the Communication Port Server configuration as well as other multi-level interfaces to external sources such as STU-IIIs as shown in Figure 9.

This example consists of a Communication Port Server with multiple interfaces to both STU-III connections and to local single level terminals, communicating over an Unclassified data network to a remote Trusted Guard which supports multiple single level LANs or Hosts. This, along with other types of configurations, will allow for cost saving and convenient configurations under which the NES platform can operate.

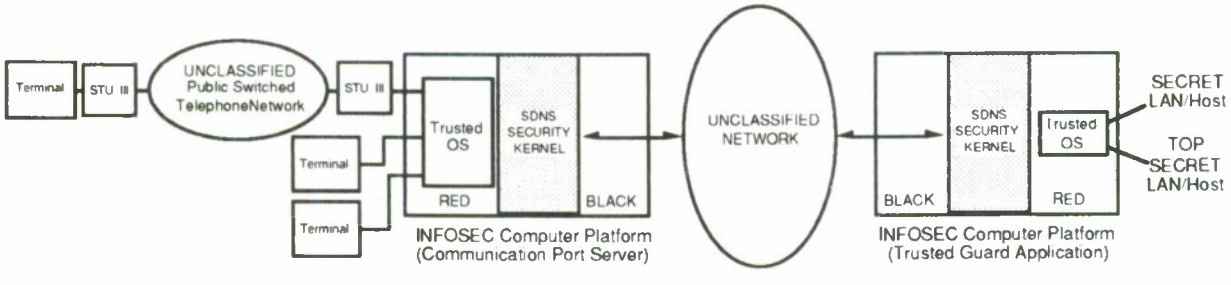


Figure 9 Communication Port Server Application

SUMMARY

In summary, combining the NES hardware with a wide range of various secure applications developed on Trusted Operating Systems, will provide cost effective and convenient methods of implementing data and communication security.

SDNS SECURITY MANAGEMENT

Wayne A. Jansen
NIST
Technology Building, A-216
Gaithersburg, MD 20899

1. INTRODUCTION

The Secure Data Network System (SDNS) program began in August of 1986 through the sponsorship of the National Security Agency (NSA). The goal of the program is to establish a communications architecture and protocols for protecting both unclassified and classified computer networks. The SDNS standards are intended to facilitate the secure interconnection of open systems within an internationally recognized framework for communications. The SDNS architecture which provides such security services as integrity, confidentiality, authentication, and access control of user data, as well as key management and systems management capabilities, is based on the International Organization for Standardization (ISO) Reference Model of Open Systems Interconnection (OSI) [1].

The SDNS program ended in 1989, and made its results available through the National Institute of Standards and Technology (NIST) [2-4]. Those documents specify security protocols (SP) at the Network (SP3) and Transport (SP4) layers of the ISO Reference Model, a message security protocol (MSP) at the application layer, an application layer key management protocol (KMP), and a framework for access control. At the conclusion of the SDNS program, it was recognized that areas of security management, outside of key management, were incomplete. A follow-on effort to SDNS, the SDNS Upgrade Program (SUP), included tasking for security management with a focus on the key management and lower layer security protocols. The SUP produced an initial set of SDNS security management documents [5-9] by September 1991. They contain a security management architecture and specifications for the elements of management information for the SDNS protocols studied, including the underlying security mechanisms and cryptographic facilities on which the protocols rely. This paper gives an overview of the SDNS security management framework and elements of security management information.

2. SECURITY MANAGEMENT IN THE SDNS ARCHITECTURE

2.1 Management Services and Protocols

In keeping with the commitment to OSI communications standards, SDNS security management builds directly upon OSI systems management. OSI systems management offers distributed network management capabilities, comprising five functional areas: fault management, configuration management, accounting management, performance management, and security management. It is the last of these functional areas upon which SDNS security management is based.

The set of OSI systems management standards includes standards for an application layer service and protocol [10,11], used to convey management information and perform management functions. The management standards also include definitions of generic systems management information [12], and guidelines for the specification of additional elements of management information [13].

OSI systems management provides mechanisms for monitoring, control, and coordination of resources within the OSI environment. A managed object class is the abstraction used within OSI systems management to represent a view of a resource. The view rendered by a managed object class consists of a set of attributes that represent characteristics and properties of a resource.

A managed object is an instance of a managed object class in which the attributes have values assigned. A managed object class can be thought of as a template for the instantiation of a managed object. The value assigned to a particular attribute (i.e., the naming attribute) allows an object instance to be distinguished from others of the same object class. All management activities are conducted through the manipulation

of managed objects. For the sake of brevity, whenever the intention is clear from the context, the term "object" is used in this paper to indicate either an object class or an instance of an object class.

A collection of managed objects pertaining to an open system is referred to as a management information base (MIB). Management applications interact according to a functional model based on asymmetric manager/agent roles. Through the Common Management Information Services and Protocol (CMIS, CMIP) [10,11], managers interact with agents to accomplish management activities. A system acting as an agent in relation to one manager, may in turn, play a manager's role with regard to other sub-agents to enable a broader span of control. Common operations for manipulating objects include the creation and deletion of objects, the getting and setting of attribute values, the evoking of predefined actions, and the sending of event reports.

Note that the security management functional area differentiates itself from the other functional areas primarily by the sensitivity of objects managed, rather than the operations used to manage them. Objects representing security services, protocols, and mechanisms can be viewed as residing within a security management information base (SMIB). Depending upon the security policy of the system, the SMIB may or may not coincide with the MIB. Security management functions include the ability to: (a) manage security objects, their operational and administrative states, and the relationships between security objects and other objects; (b) report, collect, and review security events; and (c) establish and configure security audit trails. However, only the last two functions apply exclusively to security management.

2.2 Security of Management Operations

Management operations, particularly those dealing with security management, demand a high degree of protection, due to their potential to interfere with systems operations. Two obvious ways to protect management operations are either to build protection into the management protocol that operates at the application layer, or alternatively, to have the management protocol protected by security protocols at the lower layers. The former approach is appealing for near-term implementations of CMIS and CMIP since it is self-contained and independent of the standardization progress of the lower layer security protocols. Nevertheless, SDNS security management takes the latter approach since the standards for lower layer security protocols and key management are already in place, and their use by management avoids having redundant mechanisms elsewhere in the architecture.

The SDNS architecture prescribes a two-pronged scheme for the protection of communications, relying on both the key management protocol and a lower layer security protocol. The key management protocol is a self-protecting application layer protocol, used to establish security associations for lower layer security protocols. Security associations consist of a set of negotiated security services, and associated traffic key material. All other application protocols, including systems management, use an established security association through a lower layer security protocol. The security protocol enforces the protection dictated by the security association during the communications of the application. Because a management application can manipulate sensitive information concerning the formation of security associations, including those for itself, its operations must be protected to the same degree as those of key management, and its role restricted accordingly.

The SDNS architecture requires key management functionality to be collocated with other applications within an end-system, whenever a lower layer security protocol is employed. Although collocated, the communication services afforded it are distinct to allow implementation in separate hardware components for high assurance environments. In such arrangements, key management may be considered as a hidden internal host. Figure 1 illustrates a dual stack model used to represent the partitioning of key management communication services from normal user communication services [14]. Note that because the key management protocol is the means by which keying material is provided to a lower layer security protocol entity, it must be able to bypass the lower layer protection to avoid the problem of recursion.

SDNS key management is viewed as a distinct, yet integrated part of security management. It is considered part of security management because sensitive information is manipulated. Because key management uses

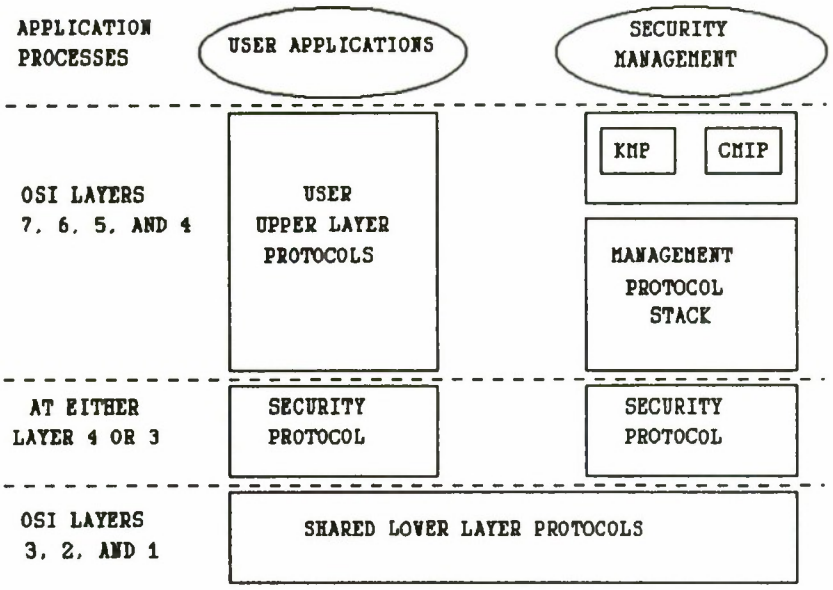


Figure 1: Dual Stack Model

its own unique protocol and its operations are a prerequisite for providing protection to management operations, it is also a distinguished subset of security management. With this perspective, the dual stack model applies equally to key management and to security management as a whole.

3. MANAGED OBJECT CLASS IDENTIFICATION

3.1 Data Model Development

The object-centric orientation of OSI systems management information modeling demands careful determination of the required object classes, their attributes, and their relationships to one another. A conceptual data model forms the basis for developing managed object class definitions. The data model gives an overview of the scope of SDNS security management information elements.

The objects identified during the modeling process fall into two general categories. The first category contains information needed by key management and the lower layer security protocols to perform required security services. The SDNS key management and security protocols provide confidentiality, integrity, authentication, and access control security services. There are also pervasive service mechanisms such as audit trail and event detection, that are implicitly provided.

The second category contains information needed to monitor and control the progress of security services and their underlying security mechanisms. This information includes system level information concerning the configuration of security protocol entities; security protocol entity information concerning the type and version of protocols that may be active in the system; algorithm information concerning the algorithm identifier and parameter values of cryptographic algorithms; and auditing information concerning the state of the system, protocol entities, and security associations.

Figure 2 illustrates an object-relationship model of the SMIB. Further associations between these security management objects and other managed objects within the MIB are not shown. Within the figure, a single headed arrow is used to indicate a one-to-many relationship between objects in that diagram, a double headed arrow indicates a many-to-many relationship, and a line without arrowheads indicates a one-to-one relationship. The SMIB objects are elaborated further in the next section. The reader is referred to the SDNS security management documents [5-9] for a more detailed discussion of specific object class attributes than that given in this paper.

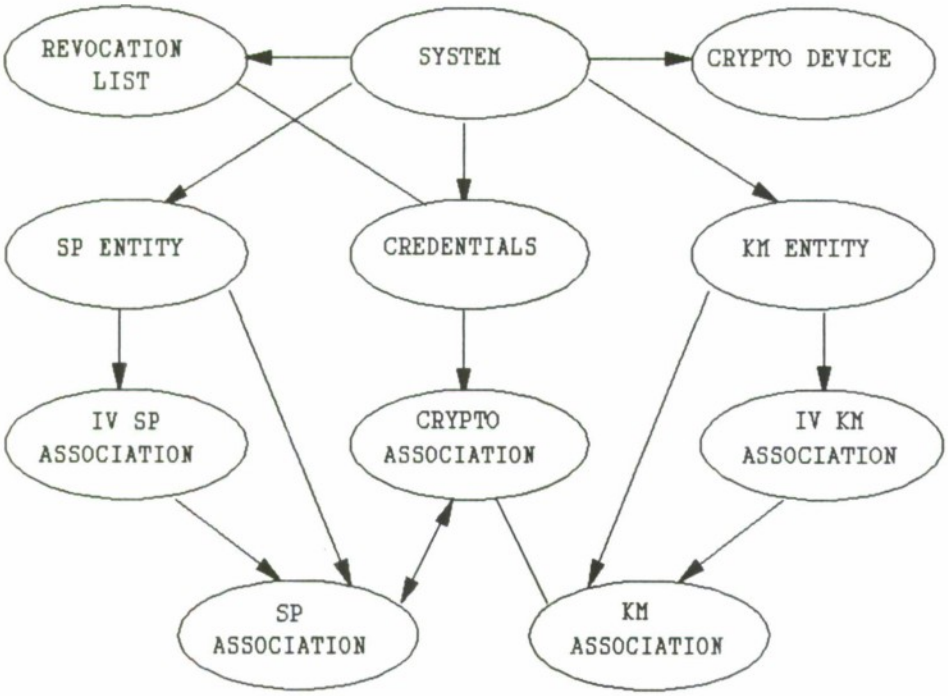


Figure 2: Object-Relationship Model of the SMIB

3.2 Object Class Descriptions

3.2.1 Identification and Authentication Objects

The objects maintained in the SMIB for identification and authentication purposes are:

(a) the system object,
(b) the credentials object, and
(c) the revocation list object.

At most, only one instance of a system object is intended to be present within the SMIB for an end-system. The system object identifies the SDNS subsystem, its administration, and security policy. A common security policy must exist before a collection of open systems may intercommunicate. The policy constraints are reflected in the values of the attributes of the remaining objects within the SMIB, that control the nature and quality of the security associations that can be established.

The credentials object and revocation list object are defined for authentication purposes. For SDNS authentication exchanges, the credentials object retains a valid certificate for the system. To corroborate peer certificates, the revocation list object retains an associated list of revoked certificates. Those end-systems wishing to intercommunicate obtain SDNS certificates signed by a mutually trusted party, the key management center. The key management center also issues certificate revocation lists. The data model allows more than one certificate to be associated with the system, if permitted by the system security policy.

3.2.2 Security Protocol Objects

The objects maintained in the SMIB for the lower layer security protocols are:

(a) the SP entity object,
(b) the initial value (IV) - SP association object, and
(c) the SP association object.

The SP entity object is used to identify each security protocol entity in the end-system and convey associated information. The SP entity represented has the responsibility for protecting an instance of communications. Identification attributes of the SP entity object include the type of protocol, the version of protocol, and the operational state of the entity. Other attributes include general utilization statistics, protocol and security error counts, and error threshold limits for notifications.

The IV-SP association object defines the envelope of interoperation that is permitted with another end-system. Its attributes indicate what security mechanisms must be in effect for an SP association to be established, in order to comply with the system security policy. The IV-SP association provides all the information needed to negotiate the security context between peer SP entities. This information includes the remote entity designator, access control permissions, the required security services, and associated cryptographic algorithm identifiers. One IV-SP association object may relate to several SP association objects.

The SP association object represents the security context established for interoperation with a peer SP entity. It retains the security protection parameters and access control restrictions negotiated for the association, used to enforce the security context. It also retains information concerning the status of the association, such as event counters and thresholds. The information within each SP association object is derived from and therefore related to only one IV-SP association object. An SP association will not be instantiated otherwise. The relationship between IV-SP and SP association objects must be maintained in a logically consistent fashion within the SMIB.

Within the context of this data model, the three SP oriented object classes are generic, since they represent only the common features of the lower layer security protocols. The use of generic object classes allows the data model to present a simple, yet accurate portrayal of the information. The generic object classes can be refined into object classes specific to SP3 and SP4 through an inheritance mechanism provided as part of the object-oriented nature of OSI systems management.

3.2.3 Key Management Protocol Objects

The object classes associated with key management (KM) are somewhat similar to those defined for the lower layer security protocols. The main exceptions are that the key management objects are not generic, and their attributes have an application layer orientation. The objects maintained in the SMIB for the key management protocol are:

(a) the KM entity object,
(b) the initial values (IV) - KM association, and
(c) the KM association object.

KM entity objects are used to identify key management protocol entities in the system and retain associated information. The key management protocol entities represented are responsible for the formation of cryptographic associations to be used by lower layer SPs. Attributes of the KM entity object include the entity name, the protocol identifier and version, the functional capabilities supported, and activity counters.

The IV-KM association object defines the envelope of permitted interoperation with a peer end-system. It provides all the information needed to establish a security context with a peer KM entity. The information includes the remote entity designator, access control permissions, the required security services, and associated cryptographic algorithm identifiers. One IV-KM association object may relate to several KM association objects.

The KM association object represents an established application association and security context between KM entities. It retains the security protection parameters and access control restrictions negotiated for the association, used to enforce the security context. It also retains information concerning the progress of the association, such as transaction counters and thresholds. The information within each KM association object is derived from, and therefore, related to only one IV-KM association object. This relationship between IV-KM and KM association objects must be maintained in a logically consistent fashion within the SMIB.

3.2.4 Cryptographic Objects

The objects maintained in the SMIB that pertain to cryptographic mechanisms employed by the lower layer security and key management protocols are:

(a) the cryptographic association (crypto association), and
(b) the cryptographic facility (cryptofacility).

The cryptographic association object contains information concerning the traffic key material formed through key management. Lower layer SPs use the traffic key material within the cryptographic facility to protect their protocol data units (PDUs). The attributes of the crypto association contain information regarding the appropriate application of keying material, the local and remote key identifiers, and the identifiers of the credentials used to form the association. Note that the data model allows a single cryptographic association to be used for multiple SP associations, or a single SP association to employ multiple cryptographic associations.

The cryptographic facility (cryptofacility) object contains information for controlling the cryptographic mechanisms that comprise the cryptofacility. The information it contains allows the status, the contents, and the cryptographic algorithms of the facility to be monitored. The attributes of this object include the number of traffic keys loaded, the identifiers of loaded credentials and traffic keys, and the identifiers and parameters of the cryptographic algorithms supported.

4. OBJECT CLASS SPECIFICATION

4.1 Degree of MIB/SMIB Separation

Object class specification efforts to date have only marginally addressed the security management area. The main reason for this is the lack of international standards for security and key management protocols. Many MIB objects are being, or have been, defined by OSI layer protocol standards groups and systems management implementor groups. Relationships between those object definitions and the ones defined for the SDNS SMIB are an important issue. A tightly coupled MIB/SMIB could allow disclosure of sensitive information through access relationships defined for non-sensitive information. On the other hand, a loosely coupled or disjoint MIB/SMIB could allow a greater degree of protection for sensitive information to be provided.

For SDNS security management, the SMIB is specified completely independently of the MIB. That is, the entire set of SDNS security management object class definitions is treated as a separate subsystem from other sets of managed object definitions that may exist, though the latter may contain seemingly related object classes. For example, the lower layer security protocol object classes clearly contain information that relates to information contained within object classes defined for the network and transport layer protocols. A protected transport connection will, by definition, use an SP association. However, if separation is maintained, a manager that accesses the MIB to determine transport connection information cannot determine the SP association being used for that connection without gaining access to the SMIB, as well.

The disjunction between the MIB and SMIB means that, for SDNS systems, security management can be made an autonomous capability with respect to the rest of systems management, perhaps requiring a higher level of authorization. This approach is consistent with the dual stack model mentioned earlier. Note too, that the ability to provide separation of the MIB from the SMIB may be disregarded in environments where it is not required by the system security policy.

4.2 Refining Generic Object Classes

Objects representing specific SP4 or SP3 entities are derived from the generic SP entity object through the inheritance mechanism. Inheritance enables the extension or specialization of an existing object class to form a new object class. In Figure 3, which illustrates the required specialization, an object at the arrow tail represents an heir of the object at the arrow head.

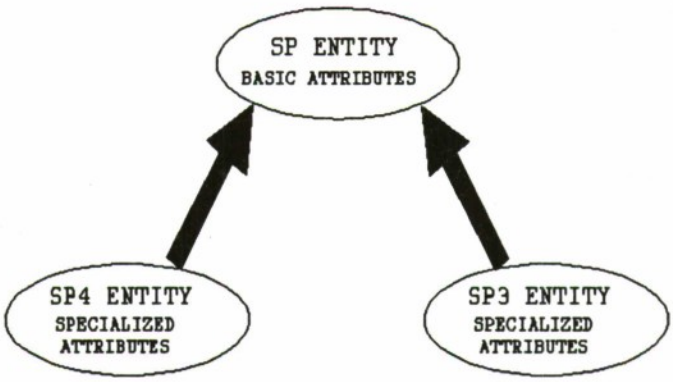


Figure 3: Specification of SP Entity Object Class

All SP entity attributes are inherited or retained by the SP4 and SP3 heirs. Additional entity specific attribute information may be defined as appropriate for the specialization. Entity specific attribute information may include the service access points serviced and/or utilized, entity specific counters and thresholds.

A similar refinement occurs for the other generic security protocol objects: IV-SP association and SP association. The IV-SP association and SP association object classes are specialized for both the SP3 and SP4 specific attributes. The main differences are in the security mechanisms available and how they are applied. For example, SP4 specialization includes attributes concerned with managing connection integrity through Transport sequence numbers.

Besides new attributes, the SP3 refinement also requires additional object class definitions. Unlike SP4, which is an extension to the Transport protocol, SP3 is an independent Network protocol. Consequently, SP3 requires a full complement of information elements similar to those defined for other Network protocols. Note that the object class definitions for SP3 are based on a revision [15] of the original SP3 standard [2]. One important difference is that the revised SP3 supports both connectionless and connection oriented network protocols, whereas the original SP3 supports only connectionless. Figure 4 gives the data model for these additional object classes, using the arrow notation defined earlier.

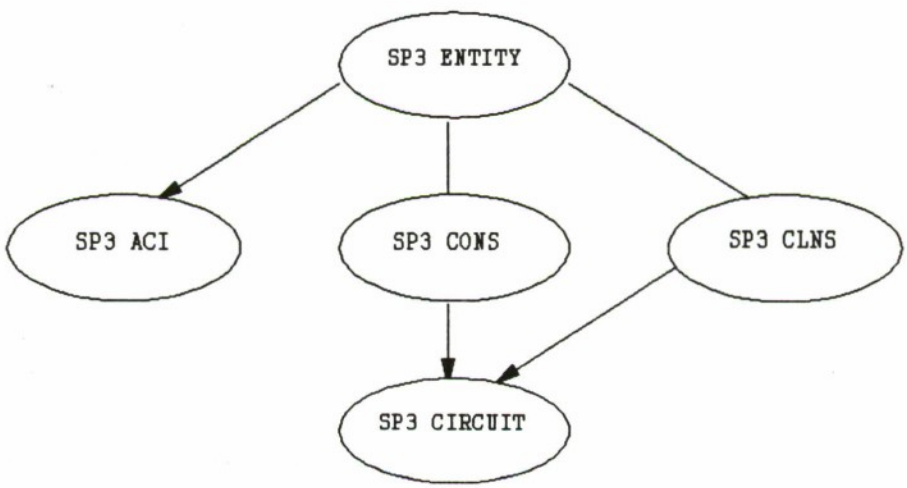


Figure 4: SP3 Additional Object Class Definitions

The additional SP3 object class definitions are: SP3 access control information (ACI), SP3 connection oriented network service (CONS), SP3 connectionless network service (CLNS), and SP3 circuit. The SP3

access control information object contains details of the end-systems protected by the SP3 entity. This information is needed for situations where the SP3 entity performs a secure gateway function. The SP3 CONS and CLNS objects serve primarily as placeholders for information regarding the type of network service. The SP3 circuit object represents an instance of communications that may be associated with either type of network service.

4.3 The Containment Hierarchy

OSI network management standards rely on a hierarchical schema called the containment hierarchy for relating object instances. This hierarchy indicates the access path to an object of a specific class. A set of name bindings is used to specify the possible relationships between a superior and a subordinate object instance. Each object class definition includes an attribute to uniquely name an object instance, relative to a superior object. Once, the name of an object instance is known, access can then be gained to the remaining attribute information for that object.

To conform to this hierarchical naming approach, the network-oriented data model of the SMIB must be recast as a hierarchy. Each object of the data model, while being retained, may now be either an object in the hierarchy, or an attribute of an object in the hierarchy. The choice depends on the type of access needed for the information. Figure 5 illustrates a hierarchical schema for the data model of the SDNS SMIB. The tail and head of an arrow are used to indicate the name binding relating a subordinate object to its superior.

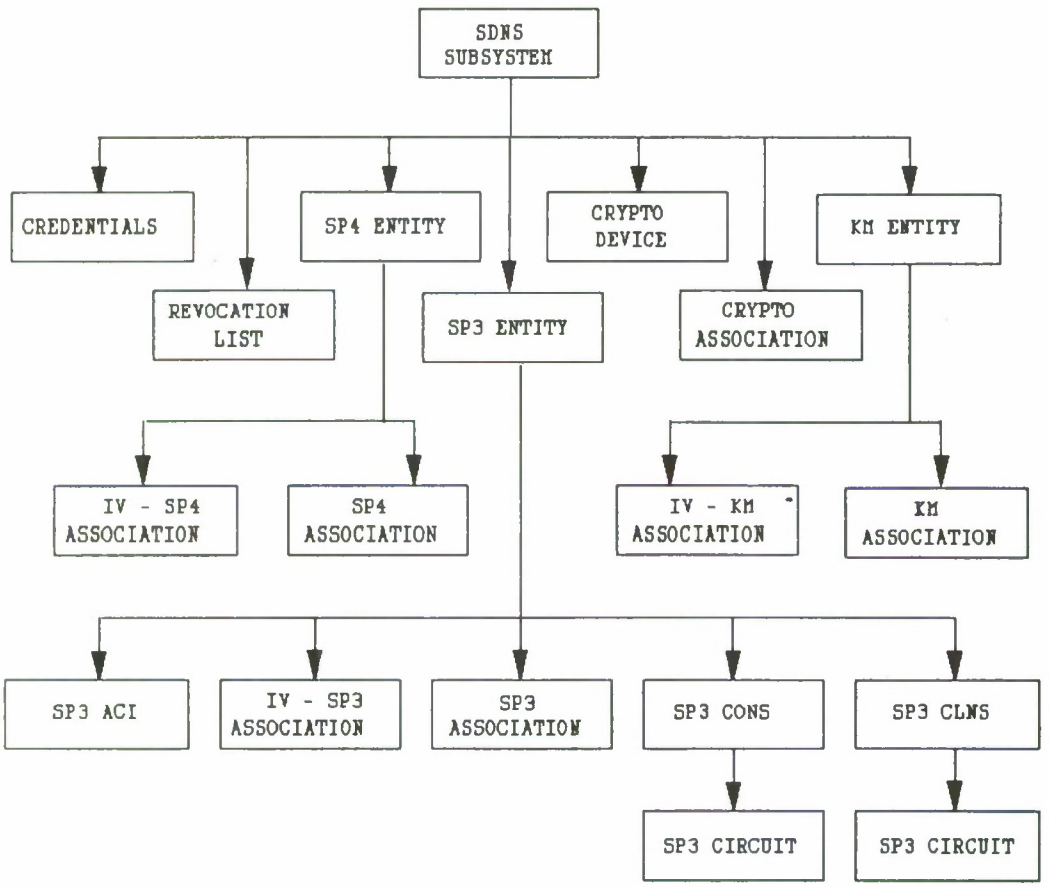


Figure 5: SDNS Security Management Naming Hierarchy

The folding of objects of the data model into attributes of other objects is possible for several reasons. The hierarchical nature of management information implies amalgamation of disparate information at

conceptually distinct levels. Since attributes are specified using the ASN.1 notation, complex relationships can be expressed without much difficulty. Some objects in the data model, because they enjoy a simple one-to-one relationship with another object, may be easily folded into that other object.

The expression of the SMIB data model as a containment hierarchy has two notable features. First, the SDNS system object of the data model is considered a subordinate object to the uppermost object in the hierarchical schema, the system object, and renamed to be a subsystem. Second, the cryptographic association object is placed at the second level of the SDNS containment hierarchy because of its many-to-many relationship with the SP association object and need for equivalent accessibility.

4.4 Incorporating Generic Object Classes

The set of OSI systems management standards includes the definitions of managed object classes that are referenced by the various systems management functions (SMFs). The definitions are generic in the sense that they support the SMFs and are intended to be used in other specifications, either directly or as a super-class, for inheritance into specialized object classes. Wherever appropriate, these SMF generic object classes are used to provide additional management capabilities. They include:

(a) event forwarding descriptor (EFD),
(b) log,
(c) access control descriptor (ACD),
(d) target access control information (ACI), and
(e) authorized initiators.

In order to use these built-in generic object classes and their associated SMFs, they must be specified as part of the SDNS subsystem. Figure 6 illustrates their positions within the containment hierarchy.

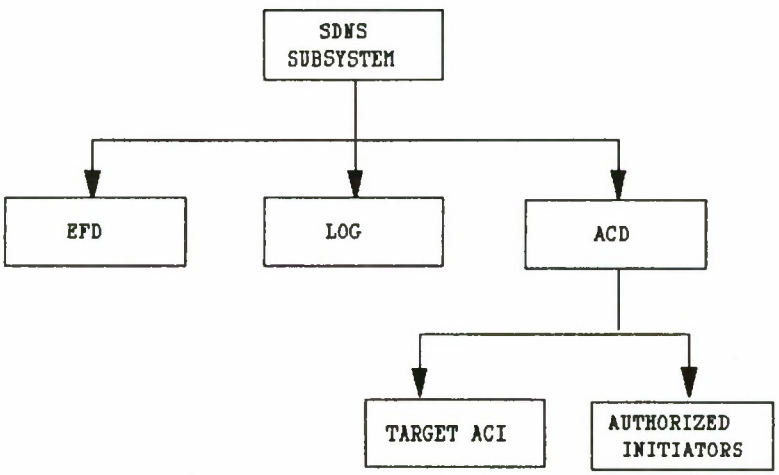


Figure 6: Naming Hierarchy for Generic SMF Objects

The EFD indicates the conditions that must be met in order to forward a local notification as an event report to a destination for logging. It also contains the destination and backup addresses for the reports, providing flexibility in controlling the levels of event reporting. The log object class contains information useful for the collection of audit data, used to control the logging of information within management protocol data units.

The ACD object class, along with the target access control information and authorized initiators, is used to control access to management information. Note that these objects are distinct from the SDNS access control objects. The ACD object class contains the access control policy, the global and default access control rules, and the item rules for the access control objects. The target ACI has the rules for protecting target object and attributes, and the names of authorized initiators. The authorized initiators object class provides the access control list, capabilities, caveats, codewords and authentication information.

5. SUMMARY

Since its inception, the SDNS architecture has been aligned with the ISO Reference Model for OSI. SDNS security management continues this commitment through observance of, and reliance on, the OSI systems management standards. The elements of management information defined for the SDNS SMIB, along with the prescribed use of an independent protocol stack for security management operations, round out the existing SDNS standards in a complementary fashion.

The SDNS security management specifications provide one of the few completed efforts in this emerging area and, as such, can provide guidance to future efforts. The defined elements of management information include object class definitions for the SDNS SP3, SP4, and key management protocols. Other defined elements concern support for cryptographic and other security mechanisms employed by the protocols. The SDNS managed objects are specified independently of objects pertaining to other functional areas. This independence provides a degree of stability for the specification and an opportunity to separate security management information from other types of management information. These features make SDNS security management a promising approach for both classified and unclassified environments.

REFERENCES

[1] ISO 7498, Information Processing Systems - Open Systems Interconnection - Basic Reference Model, 1984.

[2] NISTIR 90-4250, Secure Data Network System (SDNS) Network, Transport, and Message Security Protocols, February 1990.

[3] NISTIR 90-4262, Secure Data Network System (SDNS) Key Management Documents, February 1990.

[4] NISTIR 90-4259, Secure Data Network System (SDNS) Access Control Documents, February 1990.

[5] SDN.1001, System Security Management Architecture for the SDNS Subsystem, Revision 1.0, September 1991.

[6] SDN.1002, Elements of Management Information for the SDNS Subsystem, Revision 1.0, September 1991.

[7] SDN.1003, SP3 Elements of Management Information, Revision 1.0, September 1991.

[8] SDN.1004, Elements of Management Information for the SP4 Protocol, Revision 1.0, September 1991.

[9] SDN.1005, Elements of Management Information for the Key Management Protocol, Revision 1.0, September 1991.

[10] ISO/IEC 9595, Information Technology - Open Systems Interconnection - Common Management Information Service Definition, ISO/IEC JTC1/SC21 N5302, November 1990.

[11] ISO/IEC 9596, Information Technology - Open Systems Interconnection - Common Management Information Protocol Specification, ISO/IEC JTC1/SC21 N5303, November 1990.

[12] ISO/IEC 10165-2, Information Technology - Open Systems Interconnection - Structure of Management Information - Part 2: Definition of Management Information, ISO/IEC JTC1/SC21 N6363, August 1991.

[13] ISO/IEC 10165-4, Information Technology - Open Systems Interconnection - Structure of Management Information - Part 4: Guidelines for the Definition of Managed Objects, ISO/IEC JTC1/SC21 N6309, July 1991.

[14] Paul Lambert, Architectural Model of the SDNS Key Management Protocol, Proceedings of the 11th National Computer Security Conference, October 1988.

[15] SDN.301, SDNS Security Protocol 3 (SP3), revision 2.0, September 1991.

SECURITY MANAGEMENT:
Using the Quality Approach

Richard W. Owen, Jr.
Computer Security Official
Mission Operations Directorate
Johnson Space Center, NASA
Houston, Texas 77058

Abstract: This paper describes the process that took place when the Mission Operations Directorate (MOD) of the Johnson Space Center (JSC), NASA, transitioned from a classified, to a sensitive but unclassified, computing environment. MOD is involved in the development and operations of two major programs. Included in the paper are the organizational and total quality management (TQM) techniques that enabled the new security organization to be successful enough to receive special recognition from the NASA Administrator for avoiding approximately $30M in unnecessary costs for security.

Introduction: In the late 1980s the Mission Operations was beginning to phase down classified operations for the Space Shuttle program. It was also involved in gearing up to provide similar operations support to the Space Station program, which definitely had no Department of Defense involvement. To many, this meant that we no longer had any security requirements at all. New laws at the state and federal level and an increasingly hostile environment of hackers, viruses, and worms ensured that some form of security must remain. This paper will describe the new security program that was established for protection of sensitive but unclassified computing assets. We will begin by examining the mission, structure, and size of the Mission Operations Directorate.

Mission: The purpose of MOD is to train the astronauts to operate the Space Shuttle through various contingency situations to accomplish its mission. Please refer to Figure 1. It also trains the people that will monitor the Shuttle during its operation. Process owners, which are like users, provide inputs to the Flight Design and Computation Facility (FDCF). Here the various requirements are integrated into a mission plan. The mission parameters of the plan are then sent to the Software Production Facility (SPF). The SPF integrates the mission specific data from the FDCF with the specific Shuttle characteristics. Each Shuttle is different. The output of the SPF goes to the Shuttle Mission Training Facility (SMTF), various systems in the Flight Training and Planning Facility (FTPF), the Mission Control Center (MCC) and the Shuttle itself. The astronauts train in the SMTF for

SPACE SHUTTLE FUNCTIONAL MISSION

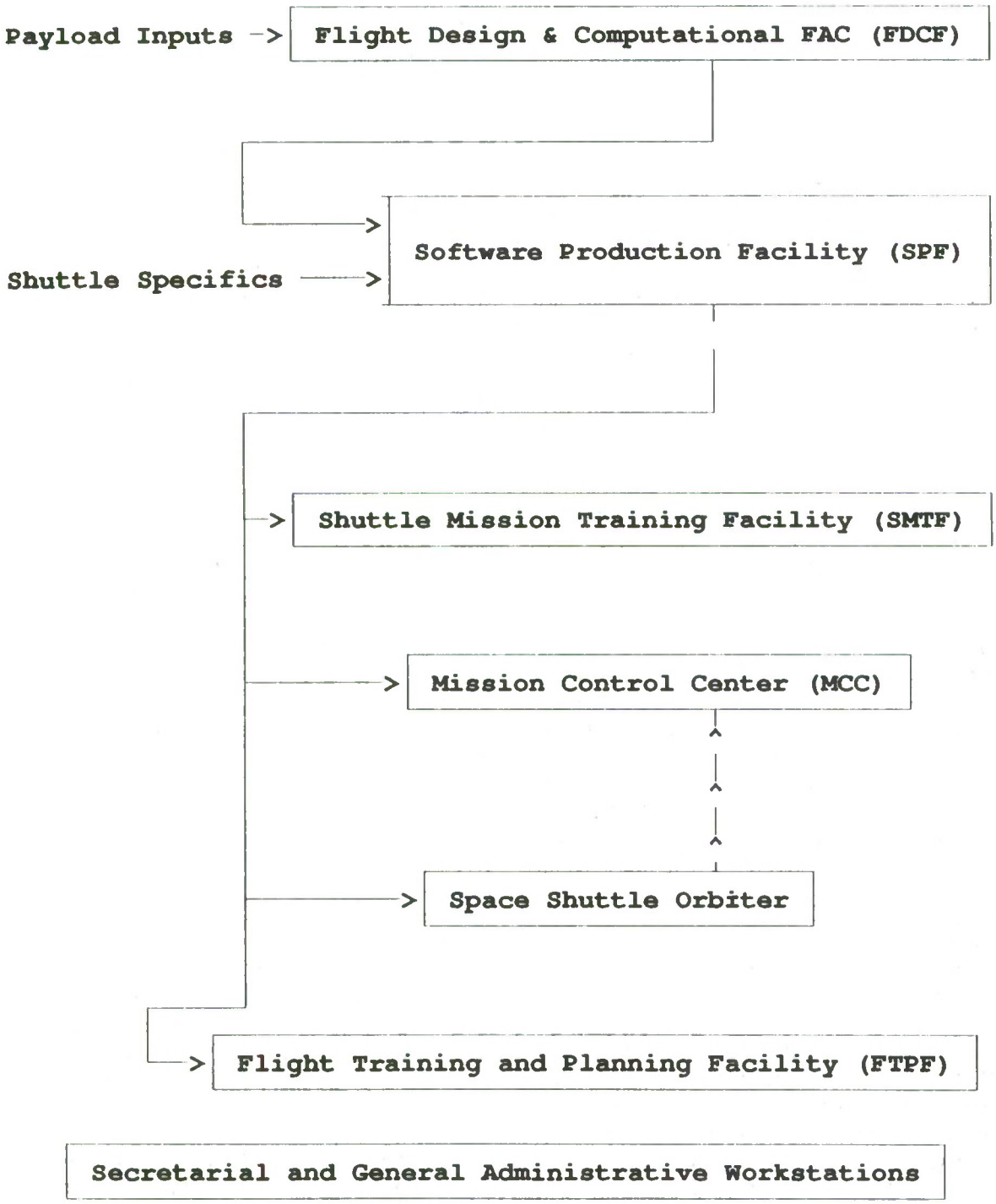


Figure 1.

SPACE SHUTTLE FACILITY SIZES

Facility Name	Numbers of			
	Lines of Code	Languages	O/S	Machines
MDF	4.1M	14	13	194
MRF	7.5M	12	2	102
MTF	2.5M	4	6	9
MMF	29.1M	21	6	151
MTPF	1.9M	6	7	7

Figure 2.

each specific mission. Training also takes place in the MCC for those who will be monitoring the performance of the Shuttle during its mission.

Figure 2 depicts the relative size of the facilities required to perform MOD's activities in support of the Space Shuttle program. The Space Station program has a larger and more complicated mission objective, but the concept is the essentially the same. The Space Station facilities are currently in the developmental stages.

Organization: The organization is composed of approximately 600 government employees and approximately 4000 employees under four major contracts. A simplified view of the structure of the organization is shown in Figure 3. The director is supported by a staff for personnel, budget, and contract administration. Reporting to the director are three assistant directors, one each in charge of the operations of each program. The third assistant director is responsible for multiple program support. The Shuttle is only considered part of MOD's responsibility for the duration of the mission. During that time, it is under the control of the Assistant Director for Shuttle Operations.

All mentioned facilities, except the secretarial and administrative workstations, fall under the Program Support sub-directorate. The facilities being developed to support the Space Station program are also in the Program Support sub-directorate. Therefore all of the managers responsible for development, operation, maintenance, and sustaining engineering report to the assistant director for Program Support.

MISSION OPERATIONS DIRECTORATE
ORGANIZATIONAL CHART

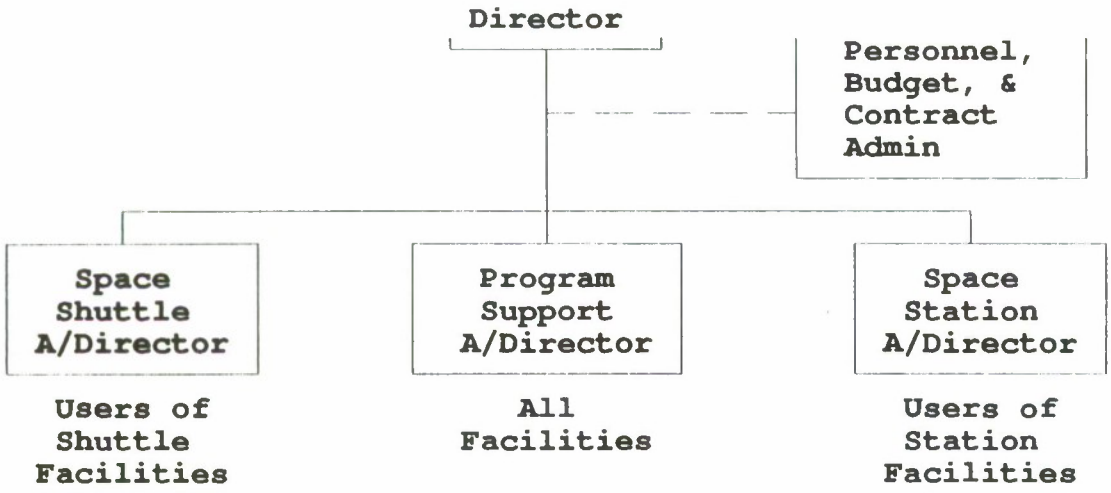


Figure 3.

Security Organization: The manager for Computer Security is called the MOD Computer Security Official (CSO). Although he is an agent of the director, he is on staff to the Assistant Director for Program Support. The relationship between the facility managers and the security personnel is shown in Figure 4. All major facilities are managed by contractors. The Space Transportation System Operations Contractor (STSOC) provides operations, maintenance, and sustaining engineering support for MOD's Space Shuttle activities. The Operations Support Contractor (OSC) provides operations and maintenance support for MOD's Space Station activities. The Training Systems Contractor (TSC) provides major upgrades to Space Shuttle training facilities and is developing training facilities for MOD's use in support of its Space Station activities. It will also provide sustaining engineering for these training facilities. The Mission Systems Contractor provides major upgrades to the mission design and monitoring facilities for the Shuttle activities. For Space Station activities they will perform sustaining engineering for the mission design and monitoring facilities which they are currently developing.

As shown in Figure 4, each facility has a contractor Facility Manager who is responsible for the overall operation and maintenance of the facility. Each facility also has a Computer Security Official (CSO). The CSO is responsible to the contractor facility manager for the security of the facility. The facility CSO is the first line of security management. He performs facility security audits and risk

SECURITY ORGANIZATIONAL CHART

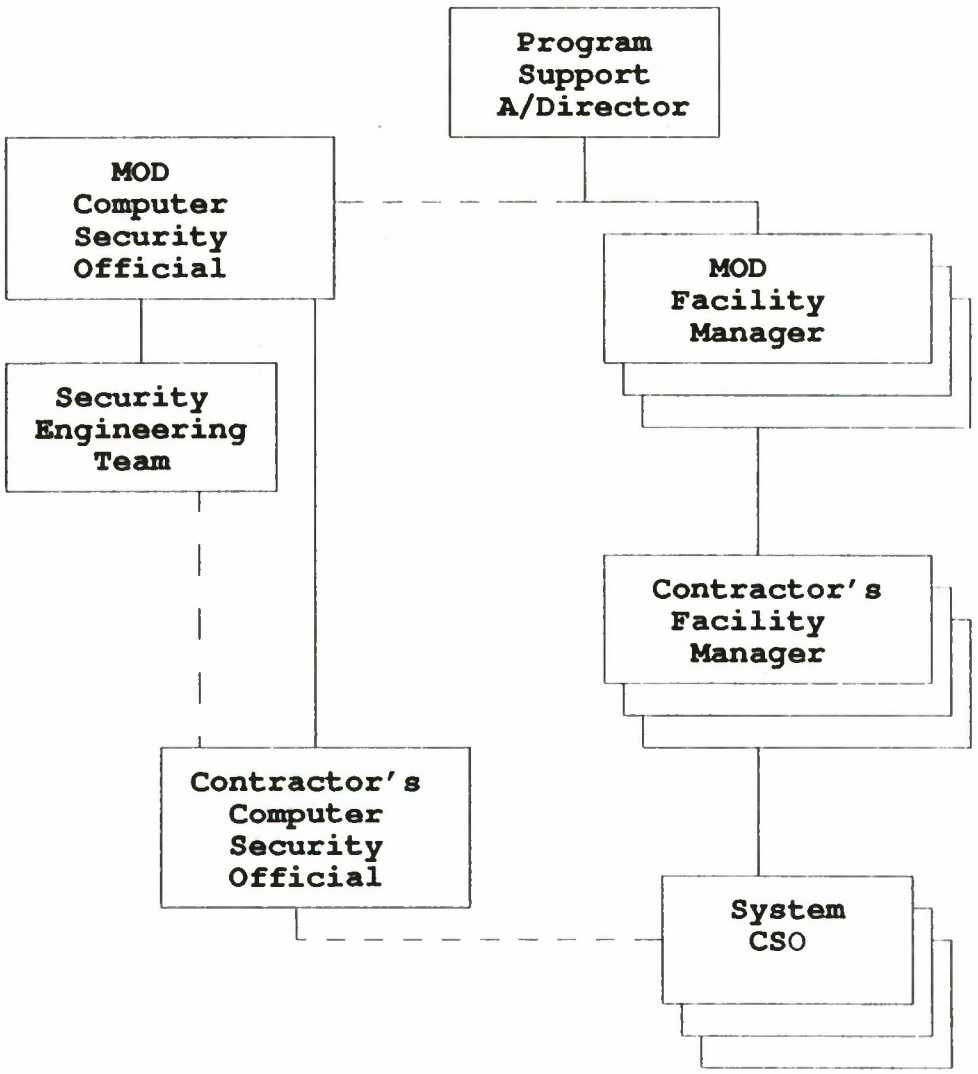


Figure 4.

analyses. Using this information he then provides guidance to the Contractor Facility Manager for needed controls or countermeasures. Each contract has a contractor Computer Security Official (CSO). He is responsible to the contract Program Manager to ensure that all security performed under that contract follows current MOD guidelines. He also directly reports to the MOD CSO. Also supporting the MOD CSO is the Automated Information System (AIS) Security Engineering Team (ASET).

AIS Security Engineering Team (ASET): The AIS Security Engineering Team is composed of senior engineers from elements of both operations contracts. Although there are

currently five current or past chapter presidents of the Information Systems Security Organization (ISSA) within the organization, for the most part members of the team have had no prior security training. They were selected because of their background in hardware engineering, software engineering, systems engineering, operations. The role of ASET is to keep abreast of the latest trends in security.

Management Objectives: In the operational plan for AIS security there are three main objectives: establish and institutionalize the AIS security process; maximize the use of technology and standards; and maximize customer support.

The establishment and institutionalization of the AIS security process must take place within all phases of the system development life cycle. This process is guided by the use of plans, policy, and procedures. Its goal is to produce and maintain accredited facilities. The objective to maximize the use of technology and standards is both a thrust at not re-inventing the wheel and at staying current. Where technology and standards are not available, the ASET uses assessments and prototyping of alternate technologies to produce physical, electronic, and administrative controls to mitigate the risk to MOD facilities. The goal of this objective is to reduce errors without impacting the workload. ASET provides training, education, and assistance to its customers, the users, operators, maintainers, sustainers and managers, to encourage their support. This results in efficient security features being built into facilities and being appropriately used to reduce operational costs and user inconvenience.

The strategic objective of the AIS security organization is to drive AIS security standards. This is being worked by our continual close working relationship with many of the standards setting organizations.

Process: The main thrust of our process is in being customer focused, managing the risks associated with the development and operations of our facilities and monitoring the metrics of how we are doing.

The following describes how we do this within MOD. By keeping current with industry, ASET recommends policy to the MOD CSO and acquires or develops tools to automate the management of security. These policies and tools are provided to the Facility CSOs to aid them in accomplishing their jobs. Based on his analyses and audit the facility CSO makes recommendations for risk acceptance or risk mitigation to the contractor Facility Manager. A course of action is then established by the contractor Facility Manager which is based on the budget constraints. This report is then reviewed by the Contractor CSO for consistent implementation within the contract and to ensure that no security issue has

been missed. After approval by the NASA Facility Manager the report goes to the MOD CSO. The AIS Security Engineering Team reviews the report and makes a recommendation to the MOD CSO. If the NASA Facility Manager and the MOD CSO cannot reach agreement on what needs to be done, the issue is elevated to the Assistant Director and/or the Director. All incidents, audits, and analyses are reviewed by ASET so that they can build a Security Management Plan for MOD. This plan contains all facility identified, unmitigated risks and all cross-facility risks. It also contains a plan for when and how risks will be reduced and all that have been accepted. By keeping the security metrics for the organization, the AIS Security Engineering Team is also able to make recommendations for improvement.

What is Working: MOD produced its first policy document, the MOD AIS Security Manual, in October 1990, the first of its kind at that level in the organization. It is currently be updated with new findings and improvements.

When the MOD AIS Security Manual was first released, all Space Shuttle facilities were evaluated for their compliance to this new policy. It was discovered that these facilities did not comply with the new policy. This was surprising since most of these facilities had just recently processed classified data. The policies and the facilities then underwent close inspection. The policies held up as being reasonable and meeting current industry practices. It was determined that it would take $32 million to correct the facilities. The ASET developed methods and tools that were tested and evaluated by the operations and development contracts. These methods and tools allowed MOD to determine the risks to its facilities. Following this exercise funds were only applied in areas where substantial return on investment could be justified. This resulted in better security than would have been provided by the $32 million and only cost $3.5 million. For this, the NASA Administrator gave the key members of the team an Administrator's Special Recognition award.

The operations contractor had gained quite a bit of experience in upgrading existing facilities with AIS security controls. The team was then given the task to produce a book on how security could be built in during the development phases of new systems. This is particularly important since we are currently developing our Space Station facilities. The first draft was released in February 1991. The final delivery of three of the five volumes is planned for September 1992.

MOD is current in the analysis of risk to its systems and applications. As these risks are identified, countermeasures are applied to mitigate the risks and/or the unmitigated risk is accepted by management. Since even the best systems and

countermeasures can sometimes go astray, plans are in place to provide for operations continunity and contingency.

The ASET is working with:

- the National Security Agency (NSA) in the area of Certification and Acceptation;
- the National Institute of Standards and Technology (NIST) in the areas of Generally accepted Systems Security Principles (GSSP) and integrity;
- the Institute of Electrical and Electronic Engineers (IEEE) on the security (DOT 6) standards for the Portable Operating System Interface for COmputer Environments (POSIX); and
- the Information Systems Security Association (ISSA) in the areas of Data Valuation and Security Architecture.

ASET, the Texas Gulf Coast Chapter of ISSA, and the University of Houston, Clear Lake produced a very successful security conference in November 1991. The theme was technology in support of integrity controls. The next conference is currently scheduled for June 1993.

Prototyping is currently underway in the area of using Artificial Intelligence and neural networking to reduce audit data. Risk Management software has been evaluated and a package selected. It is currently being configured for MOD specific concerns. Beta tests are being run on a system that will provide real time security monitoring of MVS systems. A PC-based asset control package is also being evaluated.

The most exciting accomplishment has been the paradigm shift in the attitudes of AIS security customers. Prior to the establishment of this new security organization, security had been an external entity. The Facility Manager could always ask them "What should I do?" and then do what they said with security's money. When the Facility Managers found out that this new, sensitive, but unclassified computer security program had no money, the first reaction was "no money means no requirement." Many Facility Managers thought that there was no further need for security. In fact, they did not want to hear of any security problems, especially from security people who were accustomed to telling Facility Managers that they had to stop work until they complied with a certain rule. When the Facility Managers realized that the correct functionality of their systems depended on its security, they quickly regained their interest. Luckily, by then the security people had a new perspective. The security person would now analyze the threats to the vulnerabilities of the system and recommend appropriate countermeasures. In fact, the security person now sees the Facility Manager as his customer. It is now the security person's job to find all that can happen to the system, determine the likelihood of the occurrence and recommend the most cost-effective controls

or countermeasures. With the facility CSO and contractor CSO keeping in close communications with ASET the latest in technology and procedures is always available. When there is not a fix in the market, ASET uses its contacts with the vendors to determine if a fix can be made. This paradigm shift is demonstrated in MOD Facility Managers speaking at security conferences and personally giving security briefings to their personnel.

Areas for Improvement: As with any quality project improvement must be continuous. The areas in which MOD is currently pushing for improvement are in:

- better use of inter-system integrity controls (digital signature, digital envelope),
- testing of contingency plans, ID and password reduction and
- continuing to drive the technology and standards in the needed direction.

Vision Statement: MOD will continue to push technology and standards to provide the appropriate controls to ensure that MOD's data has adequate integrity, availability, and confidentiality. This will be accomplished with minimal inconvenience to the authorized user and not one cent more than is absolutely needed will be spent on security.

Indicators of Success: This is an ongoing process of continuous improvement. MOD will continue to monitor itself for areas of improvement in keeping with its vision statement. Final success will be realized:

- when few authorized users are inconvenienced because of security,
- when the cost of security features is not identified as a stumbling block to the facility development or operations, and
- when there are so few noteworthy incidents that management must be reminded that the threat still exists.

A Security Reference Model for a Distributed Object System and its Application

Vijay Varadharajan,
Hewlett-Packard Labs., Filton Road,
Stoke Gifford, Bristol, U.K.

Abstract

This paper outlines a security reference model for a distributed environment. We consider the type of security threats and attacks faced in such an environment, and the types of security services and security mechanisms, and the management functions that are required to counteract these threats. Then we consider an "instantiation" of the the security model to a simple distributed object environment. We describe how the required security services and mechanisms can be provided, how they interact, and what are the trust dependencies involved. Finally, we conclude the paper by briefly considering a specific practical object system showing the application of the security model.

1 Introduction

Information and communication technology is on the threshhold of a new style of computing. While the last ten years have seen a focus on personal productivity, the 1990s will see the application of distributed computing to the productivity of organizations — supporting workgroups and organizations that share or trade information over a network. The trend is towards organizational distributed computing. Many styles of cooperation are possible ranging from simple communication between two entities, through the provision of services to a client by a server, to groups of cooperating entities acting together to perform complex tasks.

Security plays a vital role in the design, development and practical use of the distributed computing environment, for greater availability and access to information in turn imply that distributed systems may be more prone to attacks. Recently, there have been several well-publicized attacks on and successful penetrations of distributed systems. For instance, Robert Morris's Internet worm dramatically illustrated how open are today's systems to intruders. The widespread publicity of such events has also helped to create a greater customer awareness of security issues, and the need for protection in the technology they buy.

Figu e 1 shows a diagram of a simple distributed system architecture. It shows networks, use (people), information storage resources, computing resources, and peripherals. One may have various levels of interaction and various degrees of sharing between these entities.

At one level, we may have interaction between different resources : such as host-host, host-peripheral, and host-storage resource. At another level, one may have interactions between applications in different entities. At the outer level, we can have interactions between users.

Furthermore, the degree of interaction and sharing may also vary. At one level, there may be just transfer of information (e.g. email, edi). At another level, one may have sharing of computing resources (e.g. processors) and sharing of information storage resources (e.g. disks). At the next level, one entity (e.g. an object) may act upon another entity (e.g. object) to obtain a service from the latter. A still greater degree of cooperation occurs when entities jointly work together to perform tasks.

In each of these activities various attacks can occur and hence the need for security measures become significant. Furthermore, in practice, organizational structure and boundaries are superimposed on the distributed computing environment. The diagram shown in Figure 1 illustrates several organizations and several departments within each organization. This has impact on who has responsibility and authority over which parts of the distributed system. This in turn has implications on trust which is crucial for the placement and operation of security functions, and the interactions between them.

In addressing overall security in a distributed system, it is necessary to integrate computer system security and communication security measures to protect information both *within the system* and *between systems*. Neither one on its own can provide the required complete protection of information in a distributed environment. For instance, access controls to restrict users gaining access to a resource within a system or on a network together with suitable flow constraints to regulate the flow of information are essential. Trusted computer system mechanisms are needed to ensure the enforcement of security controls and in the provision of the necessary assurance that the correct operation of the security measures

are maintained. Secure protocols are vital to the successful operation of security measures. Security mechanisms like encryption algorithms form an essential part of the overall solution. It is the *harmonius integration* of such security features which forms the key ingredient in the development of an overall secure distributed system.

The paper is organized as follows :

- Section 2 briefly describes a security reference model for a distributed environment. We consider the type of security threats and attacks faced in such an environment, and the types of security services and security mechanisms that are required to counteract these threats. We also mention the security management functions required.
- In Section 3, we consider an "instantiation" of the the security model to a simple distributed object environment. We describe what security services are required to meet the needs, how the security services and mechanisms can be provided, how they interact, and what are the trust dependencies involved.
- Finally, Section 4 considers a specific practical case showing the application of the security model for a practical object system.

2 A Security Model Overview

2.1 Security Threats

This section describes a list of security threats that can arise in a general distributed environment (see Figure 1). In particular, we have taken into account the threat statements from various standards documents (e.g. ISO 7498-2, X.400, ECMA TR/46, DAF Security).

- *Masquerading* : This simply means the pretence of one entity to be another entity. By masquerading, an entity can get hold of privileges which it is not authorized to have in the first place. This can happen at different levels in a distributed system : within a computer system, a user or process might masquerade as another to gain access to a file or memory to which it is not authorized, while over a network, a masquerading user or host may deceive the receiver about its real identity.
- *Unauthorized use of Resources* : This includes unauthorized access to both resources on the networks as well as within a system. For instance, within a computer system, this threat corresponds to users or processes accessing files, memory or processor without authorization. Over a network, the threat may be in the form of accessing a network resource. This may be a simple network component such as a printer or a terminal, or a more complex one such as a database, or some applications within the database. Thus unauthorized use of resources may lead to theft of computing and communications resources, or to the unauthorized destruction, modification, disclosure of information related to the business.
- *Unauthorized Disclosure and Flow of Information* : This threat involves unauthorized disclosure and illegal flow of information stored, processed or transferred in a distributed system, both internal and external to the user organizations. Within a

system, such an attack may occur in the form of unauthorized reading of stored information, while over the network, the means of attack might be wiretapping or traffic analysis.

- *Unauthorized Alteration of Resources and Information* : Unauthorized alteration of information may occur both within a system (by writing into memory) and over the network (through active wire-tapping). The latter attack may be used in combination with other attacks such as replay whereby a message or part of a message is repeated intentionally to produce an unauthorized effect. This threat may also involve unauthroized introduction (removal) of resources into (from) a distributed system.

- *Repudiation of Actions* : This is a threat against accountability in organizations. For instance, a repudiation attack can occur whereby the sender (or the receiver) of a message denies having sent (or received) the information. For instance, a customer engages in a transaction with a bank to debit a certain amount from his account, but later denies having sent the message. A similar attack can occur at the receiving end; for instance, a firm denying the receipt of a particular bid offer for the tender even though it actually did receive that offer.

- *Unauthorized Denial of Service* : Here, the attacker acts to deny resources or services to entities which are authorized to use them. For instance, within a computer system an entity may lock a file thereby denying access to other authorized entities. In the case of the network, the attack may involve blocking the access to the network by continuous deletion or generation of messages so that the target is either depleted or saturated with meaningless messages.

2.2 Security Services and Mechanisms

For a particular environment, one needs to determine which threats are applicable. The overall set of security measures required to counteract the identified threats constitutes the security policy. Here we will be only concerned with logical security.

A security model should be applicable for a wide range of systems and applications, and consequently it is intended that it should include a wide range of security services that can be used and combined in different ways to meet different security policies. In particular, a distributed environment is likely to have multiple security policies and different authorities responsible for various parts of the system. We consider the aspect of multiple authorities, when we dissuss later the security management issues in Section 2.3.

In developing a security service, we need to address at least the following questions :

- what are the security information/attributes used by the service ?

- what mechanisms can be used to provide the service and what are the associated rules of operation ?

- what are the authorities that are involved in the management of the service and its associated mechanisms ?

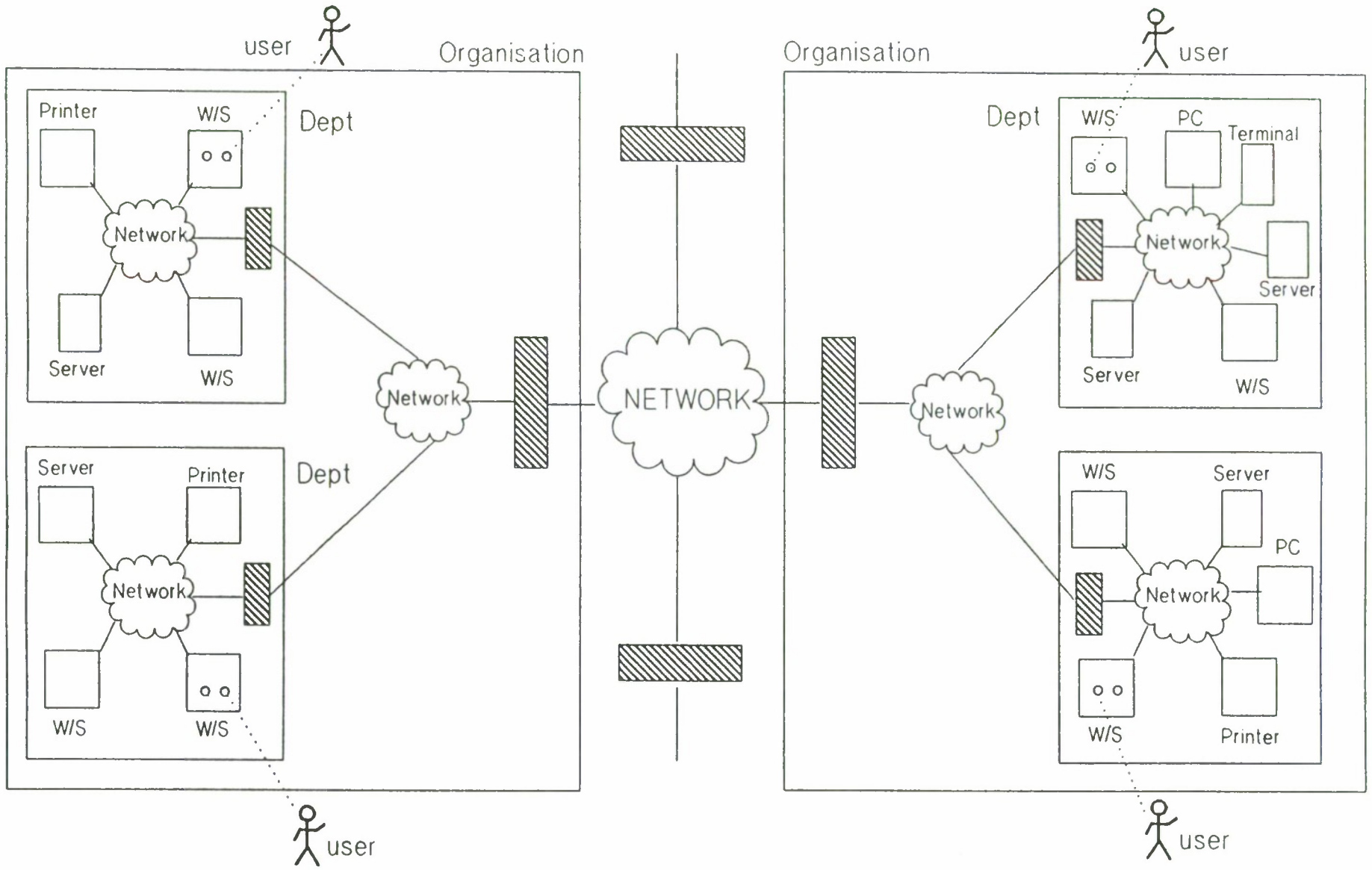


Figure 1: A Simple Distributed Computing Environment

- *Identification and Authentication* : This service provides the confidence that at the time of request, an entity is not attempting a masquerade or to mount a replay attack.

  Authentication Information : Attributes such as password, keys, smart card, retinal scan, biometrics.

  Authentication Mechanisms : Possible authentication mechanisms include :

  - Claimant presents the authentication information (such as a password) to the verifier who then authenticates it.
  - Protected exchange of authentication information, e.g. using cryptographic techniques or one-way functions.
  - Challenge-response techniques.

  Either one-way or mutual authentication may be provided. In a general situation, when two parties wish to authenticate each other, they may need to involve one or more third parties. A simple model is when we have a single trusted third party. The nature of trust between each party and the third party is an important issue in determining the assurance of the service. Examples of trusted third parties include authentication servers, key management servers and certification authorities.

- *Access Control and Authorization* : This service provides the ability to limit and control access to host systems, applications and information, and to limit what resources might do with the information contained, e.g. in applications, files. In an access control scheme, we have certain entities (initiators) attempting to access other entities (targets).

  Access Control Information : Access control information used in the decision process includes the following :

  - individual identities of initiators and targets,
  - group identities of initiators and targets,
  - security labels of initiators (e.g. clearances) and targets (e.g. classifications),
  - roles (e.g system administrator, manager) of initiators and targets,
  - the actions or operations that can be allowed to be performed on the target and
  - other contextual information, which may include time periods (over which the access is granted or denied), routing information (thereby granting an access request for only some specific routes), and location information (thereby granting access only to initiators at specific end systems, workstations or terminals).

  Access Control Rules : An access control policy essentially specifies a set of rules that define the conditions under which initiators may access targets. The decision to grant or deny a particular request is determined using access control rules and access control information associated with the request.

Traditionally, there have been two major types of access control policies : *Rule based policies* and *Identity based policies. Rule based policies* impose restrictions on all initiators. These rules form part of a mandatory access control. These controls are mandatory in that they apply to all entities and all information and the system itself has mechanisms which can enforce elements of the security policy. A common rule based access control policy is based on the use of security labels. It restricts access to resources based on the sensitivity of the target (e.g. classification) and the possession of corresponding access control information of the initiator (e.g. clearance). *Identity based policies* are based on individualized access control information such as the identity or role of the initiator. These are often referred to as discretionary in the sense that they do not enforce a set security policy, but merely permit a user or administrator to prevent access to files as the owner sees fit. Note that access control policies can also be specified in terms of groups of initiators or entities acting on behalf of initiators.

A distinction often drawn between rule based and identity based access policies is that the former is administratively-imposed whereas the latter is owner-selected. In terms of this model, the distinction lies in the control and management of access control information. A variety of choices of distribution or centralization of control is posssible in a distributed environment, ranging from a pure administratively-imposed policy to a pure user-selected one. This reflects the real world requirements exemplified by security administrators or their agents (e.g. department managers or project managers) versus individually based access control policies. Moreover, the clearances of the rule based policy and the initiator attributes of the identity based policies are essentially the same. The clearances may be considered as particular access control information associated with the initiator. Hence some of the differences between the rule based and identity based access policies are not clear cut.

Access control list mechanism is convenient when a fine granularity of access control is required and when there are few initiators. Revocation is also easier with this mechanism compared to others in the sense that revocation essentially involves appropriate modification of access control lists at the target. However this scheme is not very suitable when the initiator population is frequently changing. Capabilities are convenient when there are many initiators accessing few targets. The security label access mechanism is convenient when there are many initiators accessing many protected targets and a coarse level of granularity of access control is required.

Delegation : An important requirement that commonly arises in a cooperating environment and is associated with access control is that of delegation. The principle behind delegation is that one entity can authorize another entity "to act on its behalf". Delegation allows us to have a more flexible and dynamic form of access control. The security model can provide generic mechanisms to allow for delegation. The mechanisms allow for the following :

- From the originator, or the delegator point of view, it may wish to give only a part of its overall rights, or even just a single right. Furthermore, it may only want to grant these rights for just a limited duration. Also it should be able to identify each of its delegations so that it may at some stage attempt to revoke one of these delegations.

- The entity which has been delegated must be able to prove its delegation to an end point. It may be required for the delegated object to know the scope of its privileges, in terms of what and who they are for, and how long they may last. This can be done by passing a delegation token. It may also be necessary to set further delegations or to request access to an end point using the rights delegated to it (unless the rights have been revoked).
- The end point receiving a request from an entity claiming some delegated rights must be able to read, authenticate and verify the tokens specifying these rights and duration. The end point can then make an access decision based on knowledge of the requestor's rights together with the delegation token and the access control information.

One can provide suitable delegation schemes using the authentication and access control information and mechanisms described in [16].

- *Information Confidentiality* : Confidentiality service provides for the protection of information from unauthorized disclosure.

  Confidentiality Mechanisms : The mechanisms that are typically used to provide confidentiality are based on cryptographic techniques. In an operating system environment, it may be sufficient to protect the confidentiality of information just by using access control mechanisms. In a network environment, we can provide link-to-link or end-to-end encryption.

  - Symmetric encryption : E.g. DES
  - Public key encryption : E.g. RSA

  Confidentiality Attributes : These include :

  - Secret keys
  - Public and private keys

- *Information Integrity* : Integrity service provides for the protection of information from unauthorized modification. The unauthorized modification may involve alteration, insertion, or deletion of information.

  Mechanisms : The provision of the integrity service involves :

  (1) generation of integrity checks (at the originating end).

  (2) verification of integrity checks (at the receiving end).

  Integrity mechanisms employ cryptographic techniques to produce integrity checksums which can be used to determine whether there has been any insertion, deletion or reordering of information. This is done using feedback chaining techniques. E.g. using the cipher block chaining (CBC) technique to generate the Message Authentication Code (MAC).

- *Non-Repudiation* : The non-repudiation service provides the proof of the origin or delivery of information. This protects the sender against the threat of false denial by the recipient (that the information has been received) and protects the recipient against the threat of false denial by the sender (that the information has been sent).

  The common mechanism for providing non-repudiation relies on the use of digital signatures. Typically the digital signature is calculated using the asymmetric (public key) algorithm described above. A hash function is needed when it is necessary to sign long messages.

- *Auditing and Accountability* : The security audit service is complementary to all the security services described above in that it is not directly involved in the prevention of security violations but assists in their detection. The security audit can in turn be used to test the adequacy of the security controls and the conformance of the system with the security policy, and to assist in the formulation of any modification to the security policy.

  Auditing services are crucial both within a system and over the networks. Following audit analysis, an entity may be held accountable for its actions so that violations or attempted violations of system security may be traced uniquely to it.

  Auditing is provided by first defining the security related events and generating security audit and/or alarms. The security audit is then analyzed to evaluate the correctness and strength of the security policy. For secure reporting of audit information, auditing service may use other services such as authentication, integrity, confidentiality and non-repudiation.

- *Availability and Prevention of Denial of Service* : Denial of a service can be regarded as an extreme case of information modification in which the information transfer is either blocked or drastically delayed.

  A measure against such an attack is provided by arranging periodic exchange of information between entities to ensure that an open path exists between them. The greater the frequency of such a request response mechanism, the shorter the time period during which the denial of service attack will remain undetected. However the disadvantage is that this reduces the effective bandwidth of the network.

2.3 Security Management

When describing a security model, we also need to consider how to manage these services and how changes in policy and its enforcement can take place.

For instance, in the case of confidentiality and integrity services, it is necessary to manage the keys used in the encryption and decryption processes. In the case of access control service, we need to manage the access control information such as access control lists and the access rules. Similarly in the case of authentication, authentication information, e.g.

passwords and keys, needs to be managed. In the case of auditing, the management of audit trails and audit analysis is necessary.

In distributed systems, it is likely that there is no single authority that controls the entire environment. For instance, in an organization there may be several security managers responsible for a subset of users, objects and operations. This does not mean that it is not possible to control security in a distributed environment centrally. However even central security authorities end up trusting that the authorities responsible for local systems have implemented appropriate security.

In practice, there may be several authorities performing different aspects of these security management functions.

- Access Control and Authorization Management (Service : Access Control and Authorization)
- Authentication Management (Services : Identification and Authentication, Non-repudiation, Information Integrity)
- Key Management (Services : Information Confidentiality, Information Integrity)
- Audit Management (Service : Auditing and Accountability)

Authentication management activities typically include

- associating authentication information (e.g. passwords, keys, identities, tokens) to system entities,
- updating, modifying and revoking authentication information,
- assisting in the authentication verification process
- choosing the authentication mechanisms to be employed, and
- interaction with other security services (e.g. access control)

Access control management will typically include such activities as

- establishing and associating access control information to system entities (e.g. access control lists to files, roles and capabilities to users),
- updating, modifying, revoking access control information and rules (e.g. users "a" in role "A" with clearance C1 can access object "b" of classification C2 to perform operations "x,y, and z", for a period of time "t"),
- enforcIng access control rules (denying or granting access based on specific access control rules), and
- interaction with other security services (e.g. authentication).

Key management will be dependent on the type of encryption mechanism employed and the associated key distribution protocols. Key management will involve activities such as

- generation of suitable keys at intervals commensurate with the level of security required,
- determination in accordance with access control requirements of which entities should receive a copy of each key,
- distribution of the keys in a secure mannaer, and
- decisions involving updating of keys, deletion or removal of keys as and when they become obsolete.

Audit management activities typically include

- the definition and selection of security relevant events to be logged and/or collected,
- the enabling and disabling of audit trail logging of selected events,
- the analysis of audit trails, and
- the preparation of security audit reports.

2.4 Security Domains

One way of managing a large distributed environment is to partition it in terms of separate security domains. A security domain may be thought of as a set of elements under a given security policy administered by a single authority for some specific security relevant activities. Examples of elements of a security domain in an open systems environment include logical or physical elements such as application processes, relays, and human users. One can have different relationships between security domains. For instance, domains may have a peer-to-peer relationships in the case of two company security networks. Domains may have subdomains.

In order for interactions to take place between domains, it is necessary that there be common aspects in the security policies of the domains. Depending on the relationship between the domains, the secure interaction may come about as follows :

- For secure interaction between two independent security domains to be feasible, the security policies must enable common services and mechanisms to be selected, possibly through negotiaition, and security attributes in each security domain to be related to each other, possibly through mapping.
- In the case of security subdomains within the same security domain, secure interaction between subdomains can be established by the security domain authority.
- Secure interaction between domains via a common third party domain can be established by the common security domain for the sets of activities for which it is given jurisdiction.

In general, when developing such a multi-domain system, one needs to identify the activities involved in each domain, the role and functions of the authorities involved and the interaction between these authorities. Typically, this may involve the need for additional third parties responsible for holding some security relavant information and acting as an arbiter in resolving conflicts.

2.5 Trust

Trusted mechanisms are essential to ensure the enforcement of security controls and in the provision of the necessary assurance that the correct operation of the security measures are maintained. Whatever the combination of security services and mechanisms have been chosen to meet the security requirements and to implement the security policy, it is necessary to trust some collection of agents - whether human, software or hardware - to carry out the security functions correctly. In particular, the security management authorities mentioned above need to be trusted to maintain the security of the system.

In a multi-domain system, trust is based on mutual assurance which can be obtained either by something a domain is told or by something a domain knows. For instance, a subdomain could be told whom to trust by a superdomain. Or a domain may trust another as a result of negotiated policies.

3 A Simple Distributed Object Environment

In this section we consider an "instantiation" of the general security model to a specific distributed object environment. We consider a set of security threats, describe the security services required to counteract the perceived threats, look at how some specific mechanisms can be used to provide the services, and how they can be managed. We present specific choice of the underlying mechanisms.

The system comprises a number of objects interacting and cooperating with each other via some network medium. The objects may be of coarse granularity, such as a machine, or of a fine granularity, such as a document. We will refer to the machine objects as simply machines. Each machine may support a number of objects, and at any instant, each object is supported on a particular machine.

The objects communicate with each other by passing messages. So an object can invoke a certain operation on another object or make a request for a service from another object by sending an appropriate message. In order that the system can work effectively, a location service is required which keeps track of all the objects in the system. Furthermore suitable naming schemes will be necessary for assigning unique identifiers to the objects in the system. Some sort of name service exists as a component of most distributed object systems. For our illustration purposes here, we will not discuss these issues and assume that they are provided. We will be only addressing the security concerns.

Consider the following typical scenario : an user A logs onto a machine W and carries out certain tasks. As part of this computation, an object O1 may require some service Serv from another object O2 residing on a remote machine S.

Figure 2 shows an example of such a situation. In considering the security services required,

we first need to identify the security threats in such an environment. In fact, most of the security threats considered earlier are applicable here.

3.1 Security Threats

Typical security threats that can occur in the above scenario include the following :

1. Masquerading. This might occur for three kinds of entity :

   (a) a user A' pretending to be user A either to the local machine or to a remote machine.

   (b) an object O1' pretending to be the object O1 in sending requests.

   (c) a machine object W' pretending to be the machine W.

2. Unauthorized access to an object, e.g. an object O1 requesting a service Serv from the object O2 for which it has no authorization.

3. Threats to communication security

   (a) modifying the messages between machines W and S, thereby affecting the contents of the service request from O1 to O2, and the returned results from O2 to O1.

   (b) eavesdropping on the messages between machines W and S.

3.2 Security Services

We now consider one way of providing the security services to counteract the above threats.

We will assume that each machine object has a Security Facility (SF). For instance, this may be part of the Object Management Facility (OMF) of the machine, which is typically responsible for message delivery, object location and resource management. The SF[1] is a trusted entity and is responsible for enforcing the security policy and controls on that machine. The role and use of SF will become clear as we describe the security services and how they are provided. In particular, in the DoD TNI (Red Book) environment [2], SF can be regarded part of the Network Trusted Computing Base (NTCB).

The security services required to counteract the above threats are as follows :

1. To counteract masquerading, we need the following :

   (a) User Authentication Service

   (b) SF Trust. The local SF is responsible for identifying the source of the request.

   (c) SF Authentication Service

2. Access Control and Authorization Service

[1] In fact, OMF is also trusted.

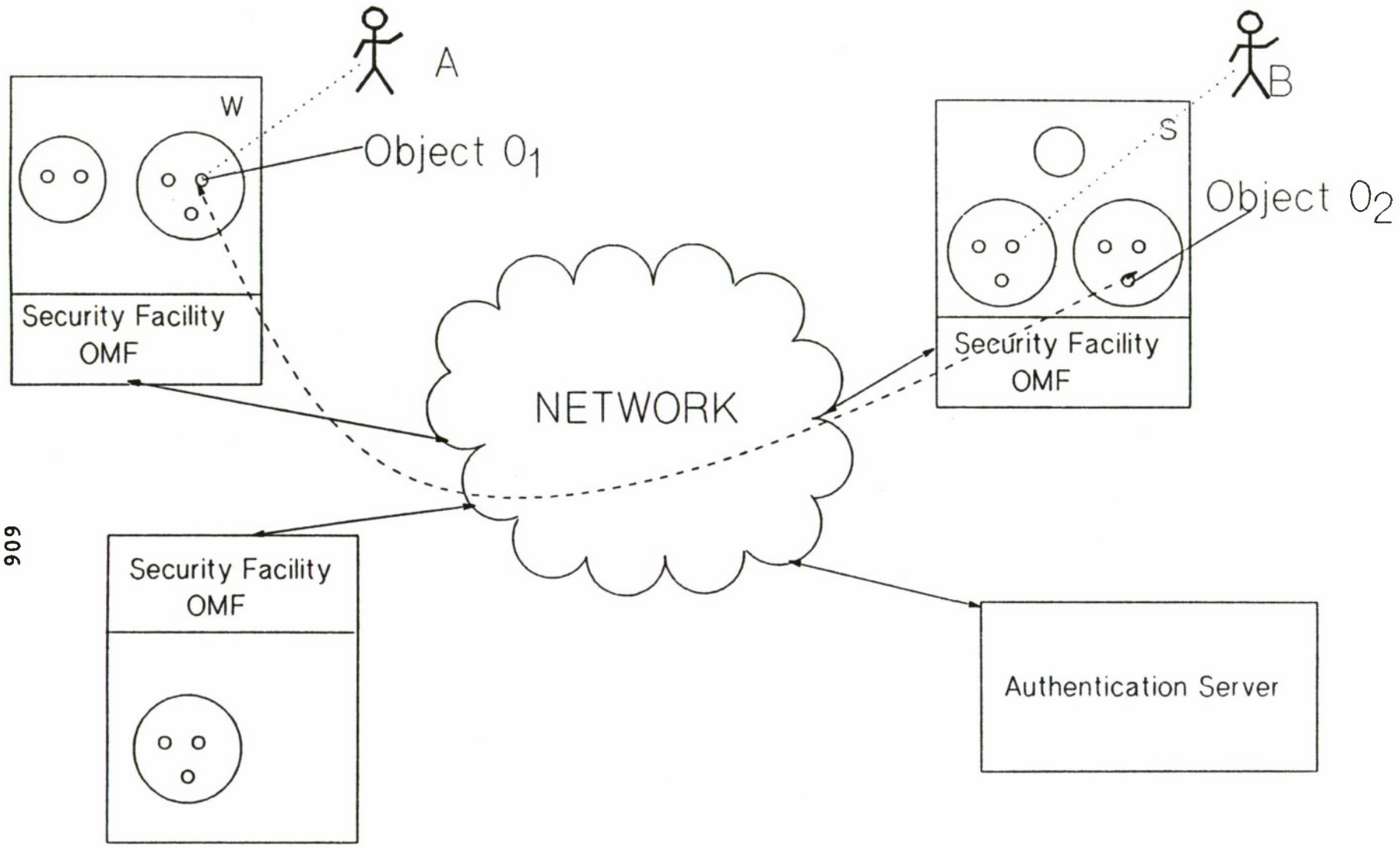


Figure 2: A Simplified Distributed Object System

3. To counteract communication threats, we need
   (a) Integrity Service
   (b) Confidentiality Service
4. Auditing Service

- **User Authentication Service** : When a user logs onto a machine, the machine's SF needs to verify the identity of that user. This is done by the user authentication service. With regard to how and where the authentication process can be carried out, we can have two cases :
  - *(i)* : The machine's SF authenticates its users locally, with all of the authentication process administered locally.
  - *(ii)* : The workstation's SF authenticates its users locally, with the assistance of an Authentication Server (AS). AS is a trusted entity on the network. One such entity per security domain is envisaged.

  Each method has its merits and disadvantages. In the case of (i), local verification of the users may be attractive from performance point of view; however problems may arise with updating of authentication information of valid users in the domain. This may be the case where any user is able to request services by logging onto any machine on the network. Case (ii) is more suitable for such a situation; however, verification requires remote communication.

  The type of authentication information used may vary from the knowledge of a password (or PIN) to possession of an intelligent token containing a detailed set of user related information, e.g. in the form of a smart card.

  Different authentication mechanisms are possible to meet this requirement : for example, the challenge-response scheme mentioned earlier. Challenge response authentication mechanism avoids the problem of sending the password in plain.

  - *case (i)* : Assume that users and machines have agreed on a publicly available one-way function to be used for authentication. The machine's SF keeps a table of user related authenticated information, e.g. keys and passwords.
    When the user tries to log in, a challenge is issued to him by the machine's SF in the form of a fresh random number. The user calculates the response by applying the publicly known one-way function and his key to the random value. The result is returned to the SF in the machine. Now the SF is able to compare this value with its own calculation, using the alleged user's key.
  - *case (ii)* : Once again, assume that a publicly available one-way function is known to all entities in the network. The AS (and not the SF) keeps a table of user related authentication information, e.g. keys and passwords.
    When a user tries to login, a challenge is issued to him in the form of a fresh random number. There are two possible schemes, depending upon the protocol chosen to provide the service.

(a) The SF in the workstation itself can issue such a challenge to the user. The user calculates the response by applying the publicly known one-way function and his key to the random value. The result is returned to the SF in the workstation. Now the SF sends the challenge and the response, together with the appropriate user information to the AS for verification.

(b) The SF in the workstation passes the user's login request to the AS and the AS issues the challenge through the SF in the workstation. The user once again calculates the response by applying the publicly known function and his key to the random fresh value and the result is returned to the AS for verification.

Such challenge-response schemes can be implemented using either the symmetric or the public key approaches.

- *Symmetric key approach* : Here the SF (case (i)) or the AS (case (ii)) has the secret keys of the users. The user employs his secret key in calculating the response to the challenge. The SF or the AS uses the alleged user's stored secret key in its verification process and compares it with the result received from the user.
- *Public key approach* : In this case, only the public keys of the users are kept by the SF (case (i)) or the AS (case (ii)), whereas the users' private keys are held only by the users or by their smart cards. The user now applies his private key to the random challenge value and the AS uses the public key of the alleged user in checking the value.

Note that Kerberos [23] provides a symmetric key approach, whereas DASS [24] uses public keys. Both these use timestamps instead of challenge-response.

In the schemes described above, the user must have the means to be able to perform the calculation on the issued challenge. This may be achieved using a smart card, given the appropriate interface to the workstation. The smart card can contain the one-way function with the user key bound in, so that it can return the response for a given challenge. (In fact, one can make use of the smart card to store additional information, such as the privileges of the user.)

Furthermore, it is necessary to protect the SFs in the machines and the AS. In the case of symmetric key approach, this table needs to be protected for both secrecy and integrity whereas in the case of the public key only their integrity must be protected.

- **SF Authentication Service** : Return to the scenario where an object O1 in a machine W makes a request for a service Serv from an object O2 on a remote machine S. The SF authentication service provides the confidence to both communicating parties that the other is the one who it claims to be. That is, both SF of W and SF of S have a guarantee that the other is as claimed.

  First consider the nature of communications between W and S. One type of communication is whereby the two machine objects setup a link which would exist for a session, allowing the exchange of several messages. Alternatively, links may be set up for the exchange of each single message. In the first case, the mutual SF authentication process can be performed once a session, whereas in the second case, authentication would be needed for every message.

  As in the case of user authentication, this service may be provided in two ways :

- *(i)* : SFs authenticates each other without the assistance of a trusted Authentication Server.
- *(ii)* : SFs authenticate each other with the assistance of a trusted Authentication Server.

In (i), each SF contains a table of keys and passwords of all the other SFs, and the SFs have agreed on a public one-way function to be used in the authentication.

One simple scheme is whereby the SFs can authenticate each other is using these keys directly. For instance, the sending SF can send a value obtained using the one-way function together with the key and the current time. The receiving SF can use the knowledge of the secret key to check the value sent. As long as the current time is within an appropriate "window", it is accepted by the receiving SF. Note that a limitation of this scheme is that replays are possible within the time "window". Alternatively, one can use a challenge-response mechanism.

In this case, when SF1 wishes to authenticate itself to SF2, SF1 first lets SF2 know of its desire to do so. SF2 responds with a challenge. SF1 now calculates the response using its key and the one-way function and returns it back to SF2. Now SF2 is able to compare this value with its own calculation, using the alleged SF1 key. Once again such a challenge response mechanism can be implemented using a symmetric key or a public key approach.

Case (i) may be simple to implement. However as before, problems may arise with respect to updating of authentication information. In general, this is less of a problem for SF authentication case than for user authentication.

In case (ii), the AS keeps a table of authentication information (e.g. keys) for all the SFs. When SF1 wishes to authenticate itself to SF2, it first authenticates itself with the AS in the usual manner. The AS now provides an authentication token to SF1 which it can use to authenticate itself to SF2. When SF1 presents this token to SF2, SF2 should have the means to verify the authentication. SF2 may carry out this verification process on its own or may involve the AS. This is dependent on the structure of the token and the protocol used. Typically such a protocol will also enable exchange of keys between the SFs to provide for communication security services. Once again, such schemes can be implemented using either a symmetric key or a public approach.

- **Access Control and Authorization Service** : The main role of the access control service in this example is to apply control over messages between objects, including the direct interactions of the user with his machine. In providing the access control service, we need to decide the following :
  - What access control information is used in the decision making process? In the implementation, this information may be implicitly stored or may be derived.
  - What access control rules are required?
  - What authorities are there in the implementation?

Returning to the scenario, we have an object O1 in a machine W with SF1 wishing to make a request on another object O2 on a machine S with SF2.

Access Control Information and Access Rules :

The access control information used in the specification of access rules can be based around some attributes of the initiating object (O1), the target object (O2), and the action being attempted.

Typical attributes of the initiator and the target objects that can be used include : the identity of the object, the class of the object, the owner of the object, the membership of the owner or object in some domain, his or her role, privileges associated with the objects, and sensitivity/integrity labels associated with the objects. As for the action being attempted, two approaches are possible. First, the decision may be based simply on the message type. Secondly, the decision may be based on more fundamental properties: the familiar "know-about", "read", "modify", "move" and "delete", and so on.

As objects are created, they must somehow have access control information associated with them, by a general default, or on the grounds of their creation in a certain context, such as a containing object. Thus, the correspondence between message type and creation is important. Furthermore in system operation, there may be a need to change the access control information and rules associated with them. These depend on the access control policy. For instance, the policy might permit some users to change the access control information to certain ob ects, whereas for some other objects, this may be totally under the control of the S' .

Using the above access control information, in this example, one can for instance specify identity-based, group and role based, label based access or capability based access control policy rules.

Also the chain of messages which cause the message and the attributes of the intermediary objects might be important. Where there is a notion of binding, where one access control decision corresponds to several interactions, this is practicable. However, checking chains of arbitrary length may be too heavy for a scheme based on decisions for every call. If this is the case, perhaps it may make sense to restrict the decision either to the immediate originator of the message or to a combination of the first and last in the chain.

Feedback to callers needs to be designed with "know-about" rights in mind. If a user tries to access an object on the remote machine, five things may happen: the object might not exist; the object might exist but fall outside the restricted view granted to the user by the owner of the owner of the remote machine; the object may not support the chosen message type; the object may not be allowed to respond the chosen message type under the restricted view; the object may be allowed to respond to a supported message type. It is desirable that the responses to the remote user in the first and second cases, and in the third and fourth cases, be the same. For the third case, this may require some discipline on the part of the developer.

Authorities

The access mediation process consists of two parts: First the access request by an object for an action on another object needs to evaluated using the relevant attributes of the objects in question and the specified access control rules. We will refer to the

entity which performs this function as *Access Decision Facility* (ADF) [7]. Secondly, the decision is enforced thereby either allowing the access request or denying it; this is carried out by an *Access Enforcing Facility* (AEF) [7]. Now the question to consider is how many such AEF and ADFs do we have for our application and how are they distributed? Several alternatives are possible, as we describe below.

First, note that as there is no notion of binding between objects, it seems necessary that access control checks need to be performed call by call. Secondly, we will assume for this example that there is no central access control authority which decides who can access what servers. Thus there are three possible locations for authority: the target SF, the initiator SF and the target object itself.

SF1 can enforce control over outgoing messages so that an initiator may not perform an action which is not in conformance with the policy for the initiator. SF2 can enforce control over the incoming messages so that the target object O2 may not receive a request that is not in conformance with the access control policy for the target. The object O2 itself may perform additional controls to decide whether to grant the request or not. In the case where both objects reside on the same machine, the controls can be enforced at the SF and/or at the target object.

Enforcing control at SF1 will help to eliminate some calls at the local end, thereby increasing efficiency. Enforcement at SF2 may be adequate for many situations whereas there may be certain circumstances where at least some of the enforcement needs to be performed by the destination object rather than the SF.

This may be the case when understanding of the semantics of the message is required in making the access decision. For instance, a given message may result in processing which creates, deletes or moves objects, under an arbitrary message type. If the access policy at the SF is described in terms of read, modify, destroy and so on, the SF will not be able to interpret the message in these terms. One possibility may be for the SF to pass on access information to objects in the form of hints, expressed in terms of read, modify, destroy and so on, and let the object to make the decision based on these. However, this will place extra trust assumptions on the object.

Alternatively, access control rules can be defined in terms of the message types. Creation of objects requires their entry in the access control scheme. This can be taken care of using the primitives provided by the SF. Given that the SF is always involved in object creation, the involvement of the local SF in access control decision can help solve the problem of the correspondence between message types and object creation.

Related to the decision of ADF and AEF placement is the problem of the location of information representing the state of access control. It may make sense to associate information on restrictions on access to an object with the object itself. However it is appropriate to store such information in the SF which is protected and trusted, whereas an object may not be.

Proxy or Delegation

An important requirement that commonly arises in cooperating object-oriented systems and is associated with access control is that of delegation. The principle behind delegation is that one object can authorize another object "to act on its behalf". Delegation allows a more flexible form of access control.

Returning to our scenario, the delegation situation occurs when the object O1 authorizes an object O3 to make a request on his behalf to object O2. Delegation is secured by verifying that an object that claims to be acting on another's behalf, is indeed authorised to act on its behalf. In practice, this also involves secure communications between objects. The delegation schemes considered in [16] are applicable for this example. In particular, the nested or linked token protocol for delegation can be used here. Once again we can implement these using either public key or symmetric key cryptosystems, as described in [16]. We will not describe these schemes here.

- **Integrity Service** : The role of this service is to ensure the integrity of messages transferred between the machines is maintained. For instance, considering our scenario, we need to ensure that the integrity of the request from the object O1 from the machine W to object O2 in the machine S is protected.

  Following the SF authentication process (cf. SF Authentication Service), this would have resulted in a key exchange between SFs which can now be used to provide the integrity service. Depending on whether symmetric key or public key method had been used, we have different possibilities.

  Symmetric key approach : First the message whose integrity needs to be protected is first hashed and then the resulting hashed (smaller) message is "signed" (encrypted) using the sending SF's key to produce a checksum. The SF at the recieving end can then calculate the checksum and see if it matches with the received one. This would not only ensure message integrity but also provide origin authentication as the sending SF's key has been used. In practice, one may include other parameters within the message such as a sequence number and a timestamp.

  Public key approach : With the public key system, the hashed message can be signed using the sending SF's secret key of the PK system to produce the checksum. The receiving SF can verify the checksum using the public key of the sending SF which it knows. Once again this provides message origin authentication and integrity. In fact, this scheme also provides non-repudiation, as no other SF could have generated the checksum. This is not so for the symmetric scheme as the receiving SF also has the ability to generate the checksum.

- **Confidentiality Service** : This service provides confidentiality of messages between the machines, for instance between W and S in our scenario. Once again, following the SF authentication process, this would have resulted in the exchange of keys between the SFs, and we can use these keys in the confidentiality service. Computational requirements may constrain us to use a symmetric key approach in the provision of this service. The messages will be encrypted by the sending SF, which can be decrypted by the receiving SF.

- **Auditing Service** : The main role of the audit service is to collect security relevant information from system operation that can be used to test the adequacy of the

security controls.

In this example, from above we see that SF in the machine plays an important role in the operation and management of security services. SF is responsible for enforcing security controls in the machine and is a trusted entity. So it is natural that SF should be involved in the auditing process.

The security policy defines those events that are security relevant. Some typical security relevant events in this example may include : request for some specific actions such as creation of new objects, deletion of objects, requests from some specific objects/machines, requests to some specific objects/machines, usage of some special objects, failure of requests, events causing changes to access control rules and access control information. For each recorded event, the audit record can identify : date and time of the event, user/object identity, type of event and success or failure of event. For creation or deletion of objects, the record can include the name of the object.

The security audit record can then be analyzed locally, or a suitably filtered version of the record can be sent to a trusted authority for further analysis. In this example, this trusted authority could be the Authentication Server.

3.3 Trust Dependencies

Some of the trust dependencies in this example include the following :

- All inter-object communications goes via SFs and the SF is trusted to convey the requests of objects it is managing accurately.
- Information stored in the SF is trusted to be secure and the SF is trusted to manage the information that it holds.
- The SF is trusted to operate correctly in the provision of all the security services.
- The SFs trust each other to operate correctly.
- It is not possible to for a user/object to be able access the representation of the object underneath the SF or at the resources of the SF itself.
- An extension of the proposed solution may allow different SFs to be trusted to different degrees. Here a service request can have the name of the originating SF marked within it, which can then be used to decide whether a service request shoud be granted.
- The user trusts the objects which are acting on his behalf not to make requests which he does not authorize.
- The Authentication Server is trusted to operate correctly. Information stored in the Authentication Server (e.g keys) is trusted to be secure and the Server is trusted to manage them.

A more general environment may have several such domains, each having for instance, one such authentication server. In such a situation, we can extend the above schemes providing

secure peer to peer interaction between these servers. This would in turn allow a user/object in one domain to request a service from another domain. For this solution, once again one can at least envisage two scenarios : In the first scenario, we have another trusted authority at a "higher level" which is responsible for managing the authentication servers. Functions of such an authority include for instance secure storage and maintenance of authentication information of the authentication servers of each domain, and secure transfer of information between authentication servers.

4 Specific Case : Cooperative Office System using Mobile Personal Computers

In this section, we briefly outline how we can use the above instantiation to provide a security solution for a specific practical case.

4.1 System Description and Object Model

Assume that there are a number of users and each user has a mobile personal computer. Typically a user stores objects such as his diary, mail, some specific documents that he may use in his computer. The user can connect his computer to a public switched telephone network, and communicate with other users and computers. For instance, he may send mail to other users or access documents from other computers.

At any one time two MobilePCs can send messages to each other via some medium, which in this note is assumed to be a public switched telephone network. It is significant that the system is largely disconnected at any time, and that communication is based on sessions. Each machine may have a number of objects; the granularity of these objects may vary. In fact, the machine itself can be regarded as an object. However when we refer to an object in this note this denotes an object residing on the machine.

Here are some characteristics of (and assumptions about) the object model.

- Each object has globally a unique identity[2].
- At any one time, a MobilePC may communicate with only one other MobilePC. During a communication session, the MobilePC has a local user — its owner — and a remote user.
- The communication is assumed to be asynchronous in nature. The object model provides transparency of location across two MobilePCs engaged in a communication session.
- The model has the concept of an *Object Manager* (OM) which is reponsible for basic message delivery, object location and resource management. Each machine has a single OM which manages all the objects present in that machine at a given time.

[2]This may not be guaranteed absolutely.

- Messages *must* pass through the OMs of the sender and the recipient. (These may be identical.) Furthermore, note that an OM can send or receive messages in its own right.
- An object belongs to an object class which determines its functionality.
- The model has both presentation and semantic objects. As far as the OM is concerned, the difference lies in a semantic object's ability to be deactivated and re-activated.

4.2 Security Threats and Services

User A uses his MobilePC to dial user B's MobilePC. After the connection is established, two users can communicate with each other and manipulate each other's objects. The types of threats in such an environment are as follows :

1. Masquerading, that is, one entity pretending to be a another entity. This might occur for three kinds of entity:

   (a) A user might pretend to be another either to a local MobilePC, or a remote user and MobilePC.

   (b) An object A sending a message to another object B might pretend that it is object C.

   (c) A remote machine which is not a MobilePC might masquerade as a MobilePC. or an Object Manager might pretend to be another.

   To counteract masquerading, we need the following.

   (a) A user authentication mechanism. (The MobilePC authenticates the user when he or she uses (logs into) the MobilePC.)

   (b) OM trust. The local OM which is responsible for identifying the source of a message.

   (c) An OM to OM authentication mechanism.

2. Unauthorized access to an object.

   In order to address this threat, we need to know how various kinds of access to objects are described, and how rights are then defined. For instance, in the case of the former, access may be described in terms of common affects of a message on an object, such as read, write, move and so on, or can be described directly in terms of message types. For the latter, rights may be given to users, to calling objects, or to objects acting for a user. Furthermore, we may need the notion of delegation whereby one object is acting on behalf of another.

   It must be stressed that not all access control schemes can be implemented on an arbritrary base. For an unprotected OM, it makes sense only to specify rights for users. This is under the control of the user (owner) of the MobilePC.

3. An entity may try to "hijack" a communication session between two other objects once it has been established.

   It will be seen below that for a protected OM, this threat is addressed through correct implementation of the object, while for the unprotected OM, this threat is subsumed in a larger threat, whereby the object model is not enforced.

4. Message modification.

   Here, a message in transit (between MobilePCs) is modified thereby affecting the received contents of the message or the apparent origin of the message.

   This threat may arise from the communications medium or from within a MobilePC itself. It may be felt that the former case is relatively unlikely: below, we propose message authenticity services which may be included in MobilePC, but would be optional services for use between communicating pairs of objects.

   For the latter case, the threat is addressed by enforcing the object model. Thus, for an unprotected OM, this cannot be achieved.

5. Message eavesdropping.

   The same comments apply for secrecy as for authenticity, above.

In considering the security services for MobilePCs, one can have two cases : an unprotected OM and a protected OM.

By unprotected OM, we mean that a user of the MobilePC is able to access the representation of objects underneath the OM or at the resources of the OM itself. So in this case, the user of the MobilePC has complete control over the OM's behaviour. Hence the only security services that one can provide here are based on those specified by the user of the MobilePC himself. As MobilePC is a single user machine (single "owner"), these security services can be used to protect the user ("owner") from other users.

1. Authentication of the local user to the MobilePC.
2. Authentication of the remote user to the MobilePC.
3. User-based access control.

In the case of the protected OM, we can make the following trust assumptions :

- It is not possible for the user of the MobilePC to be able to access the representation of the object underneath the OM or at the resources of the OM itself.
- The OM is trusted to accurately convey the requests of objects that it is managing.
- Information stored in the OM is trusted to be secure and the OM is trusted to manage the information that it holds.
- The OM is trusted correctly to identify the sending object.

The security services that can be provided in this case include the following :

1. Authentication of the local user to the MobilePC.
2. OM – OM authentication.
3. Object-based access control.

Additional Services which are required include :

4. Message authenticity and integrity, against wire-tapping.
5. Message confidentiality, against wire-tapping.

4.3 Incorporation of Security in MobilePC in Stages

In practice, one might consider the incorporation of security services in MobilePC in several stages, depending upon the relative importance of the perceived threats. One such possibility is as follows :

- Stage 1
  Unprotected OM - All three security services and a simple message integrity service.
- Stage 2
  Protected OM - All three security services and a simple message integrity service.
- Stage 3
  Protected OM - All three security services and the additional security services.

From security point of view, Stage 3 is the best option.

5 Summary

In this paper, we have outlined a security reference model for a distributed environment. We considered the type of security threats and attacks faced in such an environment, and the types of security services and security mechanisms, and the management functions that are required to counteract these threats. Then we described an "instantiation" of the the security model to a simple distributed object environment. In particular, we looked at how the required security services and mechanisms can be provided, how they interact, and the trust dependencies involved. Finally, we concluded the paper by briefly looking at a specific practical system where such a model can be applied.

References

[1] Department of Defense, Department of Defense Trusted Computer System Evaluation Criteria, DoD 5200.28-STD, Dec.1985.

[2] National Computer Security Centre, Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria, NCSC-TG-005, Vers.1, July 1987.

[3] International Organization for Standardization (ISO), ISO 7498 : Part 2 - Information Processing Systems - Open System Interconnection - Basic Reference Model - Security Architecture, 1988.

[4] International Organization for Standardization (ISO), ISO 7498 : Information Processing Systems - Open System Interconnection - Basic Reference Model, 1984.

[5] International Organization for Standardization (ISO), ISO/IEC JTC 1/SC 21/WG1 : Security Framework Overview, Working Draft, 1991.

[6] International Organization for Standardization (ISO), ISO/IEC JTC 1/SC 21/WG1 : Authentication Framework, Draft International Standard DIS 10181-1, 1991.

[7] International Organization for Standardization (ISO), ISO/IEC JTC 1/SC 21/WG1 : Access Control Framework, Working Draft, 1991.

[8] C.C.I.T.T, X.509 - The Directory : Authentication Framework, 1989.

[9] C.C.I.T.T, X.500 - The Directory : Overview of Concepts, 1989.

[10] European Computer Manufacturers' Association (ECMA), Security in Open Systems - A Security Framework, Technical Report ECMA TR/46.

[11] European Computer Manufacturers' Association (ECMA), Security in Open Systems - Data Elements and Service Definitions, ECMA 138.

[12] European Computer Manufacturers' Association (ECMA), Authentication and Privilege Attribute Security Application, Working Draft 1991.

[13] R.Rivest, A.Shamir and L.Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, Vol.21, No.2, pp 120-126.

[14] M.Gasser, A.Goldstein, C.Kaufman, B.Lampson, *The Digital distributed security architecture*, Proc. of the 12th National Computer Security Conference, 1989.

[15] Vijay Varadharajan, Stewart Black, *A Multilevel Security Model for a Distributed Object-Oriented System*", Proceedings of the IEEE Sixth Annual Computer Security Applications Conference, USA, Dec.1990.

[16] Vijay Varadharajan, Phillip Allen, Stewart Black, *An Analysis of the Proxy Problem in Distributed Systems*, Proc. of the 1991 IEEE Symposium on Research in Security and Privacy, 1991.

[17] Vijay Varadharajan, *A Security Architectural Framework for a Distributed Environment*, Proc. of the International Computer Security Conference, COMPSEC'91, London, Oct.1991.

[18] NBS FIPS PUB 46, Data Encryption Standard, U.S. Department of Commerce, National Bureau of Standards, 1980.

[19] NBS FIPS PUB 81, DES Modes of Operation, U.S. Department of Commerce, National Bureau of Standards, 1980.

[20] D.E.Bell and L.J.LaPadula, Secure Computer Systems, ESD-TR-73-278 (Vol.I-III), Mitre Corporation, Bedford, MA, April 1974.

[21] European Commission, Information Technology Security Evaluation Criteria (ITSEC), Vers.1.1, Mar.1991..

[22] IEEE, 802.10 : Standard for Interoperable Local Area Network Security (SILS) : Part A - The Model, 1989, and Part B - Secure Data Exchange, 1990

[23] J.Kohl, B.C,Neumann, J.Steiner, *Kerberos*, Vesrion 5, Draft RFC, Project Athena, MIT, Dec.1990.

[24] Charles Kaufman, DASS : Distributed Authentication Security Service, Network Working Group, Internet Draft, 1991.

[25] C.C.I.T.T, X.435 - Recommendations for security in X.400. 1988.

[26] C.C.I.T.T, SGVII Distributed Applications Framework Security Infrastructure Document, Draft, Nov.1990.

[27] International Organization for Standardization (ISO), ISO/IEC JTC 1/SC 21/WG6 : OSI Upper Layers Security Model, Draft (CD), 1991.

[28] International Organization for Standardization (ISO), ISO/IEC JTC 1/SC 21/WG : OSI Lower Layers Security Model, Draft, 1991.

[29] Open Software Foundation, Distributed Computing Environment Overview, 1990.

SECURITY WITHIN THE DODIIS REFERENCE MODEL

Brian W. McKenney

The MITRE Corporation, 7525 Colshire Drive, McLean, VA 22102-3481

ABSTRACT

The Defense Intelligence Agency (DIA) has developed an information system framework that is intended to provide a foundation for the development of architectures and implementation/transition plans, and the selection of standards and products that will meet the goals and objectives of the Department of Defense (DOD) Intelligence Information System (DODIIS) community. The DODIIS Reference Model (DRM) will be used by DODIIS planners and engineers responsible for the procurement of hardware and software associated with the upgrade of existing intelligence capabilities and the implementation of new capabilities at DODIIS sites. Security is an essential element of the DRM. This paper[1] provides a snapshot of a current task associated with updating the security-relevant portions of the DRM, presents a brief overview of the DRM, and discusses security for specific service areas defined in the model. Since the model is based on standards, this paper also identifies current and emerging DODIIS security standards for specific service areas of the DRM.

1.0 INTRODUCTION

As the DODIIS community faces declining budgets, the community will be forced to reduce duplicative system developments and life-cycle maintenance requirements to achieve cost savings. The DODIIS community must strive to integrate as much commercial off-the-shelf (COTS) software as possible to reduce the software development and operations and maintenance (O&M) costs. The community must also increase resource sharing, evolve toward "open systems," and develop common capabilities. In order for these strategic objectives to succeed, the DODIIS community must have a consistent standards-based architecture across the community.

The DIA has developed an information system framework [1] that is intended to provide a foundation for the development of architectures and implementation/transition plans, and the selection of standards and products that will meet the goals and objectives of the DODIIS community. DODIIS is "the aggregation of DOD personnel, procedures, equipment, computer programs, and supporting communications that support the timely and comprehensive preparation and presentation of intelligence and intelligence information to military commanders and national-level decision makers" [2]. The framework, promulgated by the DODIIS Management Board (DMB), provides a basis for selecting open system standards. The DMB, composed of senior DODIIS managers from DIA and the military components, is responsible for developing and implementing DODIIS policy and procedures. The *DODIIS Reference Model for the 1990s* [3], a companion document to the framework, describes the current and future standards that have been selected for DODIIS. The DRM will be used by DODIIS planners and engineers responsible for the procurement of hardware and software upgrades for existing intelligence capabilities and the implementation of new capabilities at DODIIS sites. Major goals of the DRM are to forge a consensus on the definition of an architectural model and to establish a common vocabulary and "information technology roadmap" within the DODIIS community. When applied, the DRM will help to promote interoperability and portability between DODIIS programs and applications, and reduce resources through the use of common, interoperable, and shared services.

Security is an essential element of the DRM. This paper provides a snapshot of a current task associated with updating the security-relevant portions of the DRM, presents a brief overview of the DRM, and discusses security for specific service areas defined in the model. Since the model is based on standards, this paper also identifies current and emerging DODIIS security standards for specific service areas of the DRM.

---

1 This paper was supported by DIA under contract DAAB07-91-C-N751. The publication of this paper does not indicate endorsement by DIA or The MITRE Corporation, nor should the contents be construed as reflecting the official position of these organizations.

1.1 ADDITIONAL REFERENCE MODEL DEVELOPMENTS

The National Institute of Standards and Technology (NIST) is responsible for promulgating Federal Information Processing Standard (FIPS) Publications (PUBS). A recent NIST publication is the *Application Portability Profile (APP) - The U.S. Government's Open System Environment Profile* [4]. The APP is intended to be the United States (U.S.) Government's Open System Environment profile. An Open System Environment (OSE) encompasses the functionality needed to promote interoperability, portability, and scalability of computerized applications across heterogeneous information technology platforms. An OSE extends the Open Systems Interconnection (OSI) concept [5] to the broader problems of application portability and interoperability. NIST has promulgated the APP in an attempt to integrate standards and to assist Government agencies in making informed acquisition choices regarding the selection and use of OSE specifications. The APP is a guideline directed toward managers and project leaders with responsibilities for procuring, developing, and operating information systems that support heterogeneous application platforms. As a result, the APP should *not* be regarded as a standard since the APP is not designed to cover all information system environments or meet all user requirements and needs.

The DRM is based on the NIST APP, and a goal of DODIIS is to conform to the NIST recommendations where possible. Like the NIST APP, the DRM is a guidance document. The DRM applies to upgrades of any DODIIS site hardware and software. An upgrade ranges from selective upgrades of subsets of a system at a site to a complete change of a system at a site.

The Center for Information Management of the Defense Information Systems Agency (DISA) has also developed a technical reference model that defines a target framework and profile of standards. The *Technical Reference Model for Information Management* [6] provides technical guidance to DOD components for the acquisition, development, and support of DOD information systems and associated infrastructure systems. Much of the information within the Technical Reference Model (TRM) is derived from the NIST APP and the DRM. Version 2.0 of the TRM, scheduled to be published in late 1992, will address the services and standards needed to support DOD's distributed computing requirements. Other areas to be addressed in future editions of the TRM include services and standards for tactical systems, imagery, and multimedia data transfer.

The Institute of Electrical and Electronics Engineers (IEEE) is currently drafting a Portable Operating System Interface for Computer Environments (POSIX) OSE guide that maps existing and emerging standards onto the general requirements of a complete information system. The guide is a product of the IEEE POSIX standardization effort, although its scope is much broader than the IEEE POSIX standardization efforts. The POSIX OSE Guide [7] will present a framework that identifies key information system interfaces involved in application portability and system interoperability, and will describe the services offered across these interfaces. In addition, the guide will consist of a reference model, service definitions, standards, and profiles. It is expected that future versions of the NIST APP, DRM, and the TRM will align with the POSIX OSE guide.

1.2 DRM STANDARDS AND SELECTION CRITERIA

The DRM is based on standards and is evolutionary in nature since not all of the needed standards (e.g., security standards) have been developed. The model does not mandate a specific system design, but is instead a set of standards and guidelines to be applied to all DODIIS programs and site architectures.

Standards are organized within the DRM into three categories: (1) formal/government standards, (2) de facto industry standards, and (3) other standards. Formal/government standards are designated by standards-setting or standards-approving bodies such as the International Organization for Standardization (ISO), American National Standards Institute (ANSI), IEEE, and NIST. Military Standards (MIL-STDs) also fit in this category. De facto industry standards are specifications or products that have achieved a high degree of acceptance within industry and have been implemented in numerous commercial products. Examples of de facto standards include Sun Microsystems Network File System (NFS) and the Massachusetts Institute of Technology (MIT) X Window System. The other standards are specifications, products, or program-related products that do not fit in the other categories of standards. Many of these standards are unique to the DODIIS community. Examples include the DODIIS Network Security for Information Exchange (DNSIX) and the Compartmented Mode Workstation (CMW).

Criteria for the selection of standards, products, and systems for DODIIS, currently section 3 of the DRM, will be enhanced and released as a companion document to the DRM. These criteria are intended to be used as part of a consistent approach to selecting standards, products, and systems. Criteria to be used to select standards include functionality/completeness, compatibility with other standards, maturity/stability, product availability, and supportability.

1.3 DODIIS DOCUMENTATION TREE

The *DODIIS Document Management Plan* [8] specifies the documents required to provide technical guidance to DODIIS sites for achieving an open systems environment. After the DODIIS Framework and DRM were adopted by the DODIIS community, it became apparent that a series of documents was needed to be developed that defined the future direction for DODIIS and to provide guidance to DODIIS sites and organizations supporting DODIIS. Most of the documents will establish overall DODIIS policy, procedures, or architectural guidance. The collection of documents is referred to as the "DODIIS Documentation Tree" and includes compliance documents, guidance documents, and DODIIS site documents. The DODIIS site documents must be produced by each DODIIS site to achieve the open system environment envisioned by the DRM.

2.0 DODIIS REFERENCE MODEL

Figure 1 illustrates the DRM and is a top-level representation of the components of an information system. The model is based on the NIST APP OSE Reference Model. The principles behind the DODIIS model are as follows:

- Software services are independent of hardware.
- All services interact with the operating system.
- Distributed client-server requirements will be accommodated through interactions between other service areas and network services.
- Security services are integrated within each service area.
- System management and security apply to all aspects of the information system.

The model consists of eleven major components in which each of the services are further defined by layered sets of functions. Although not illustrated within the model, an overall system security policy governs all of the components of the information system. A system security policy defines what it means for an information system to be "secure" and provides significant input to the design of a secure information system.

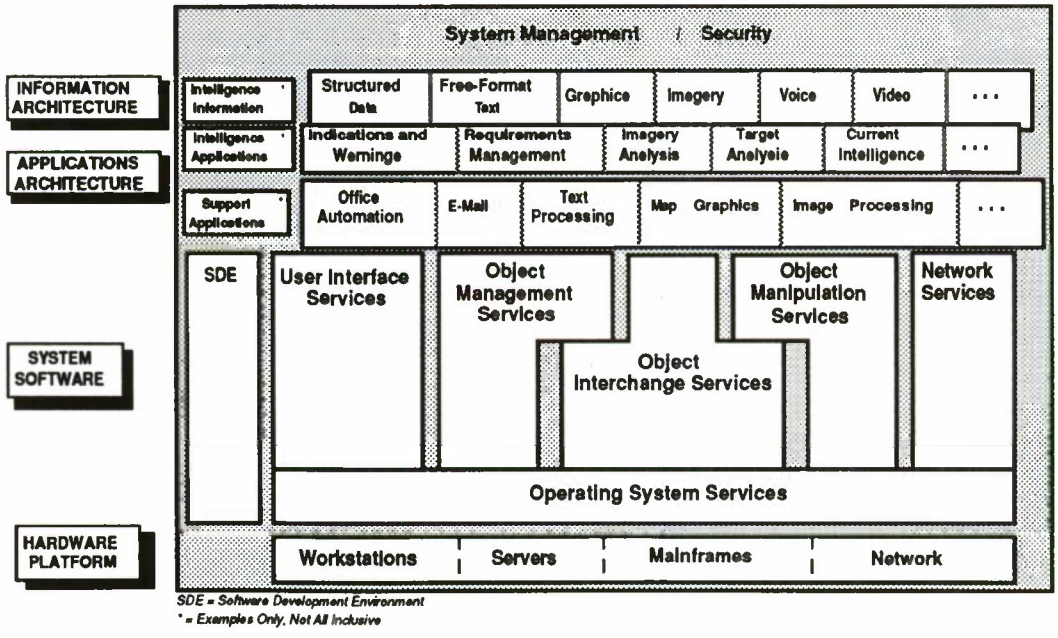


Figure 1. DODIIS Reference Model

Security services are integrated within each service area. A security service is a service that satisfies a security policy or specific security requirement. Security services associated with one of the service areas may depend on security services offered by a different service area. For example, data management security services (associated with object management services) may depend on operating system security services. A security mechanism is used to implement a single security service or a set of security services. For example, access control lists (ACLs) is one security mechanism that can be employed to implement access control within an operating system. Encipherment is an example of a security mechanism that can be used to implement a set of network security services (e.g., data confidentiality, data integrity, and authentication).

3.0 DRM SECURITY SERVICES AND STANDARDS

Currently, DODIIS standards for security can be recommended in operating system services, network services, user interface services, object management services, object interchange services, and system management services. A discussion of the applicable security services for these specific service areas, along with associated DODIIS security standards (where they exist), is provided in the following sections. Although security services and support mechanisms are associated with hardware platforms (e.g., TEMPEST, physical security, domain isolation), this paper will not address security-related services and standards associated with hardware platforms.

3.1 OPERATING SYSTEM SERVICES

Operating system services include the low-level services that manage hardware resources and the high-level software mechanisms that facilitate user and application interaction with the hardware and operating system. Operating system services include process and task management services, file-oriented services, kernel operations, commands and utilities, memory management services, and operating system security services.

3.1.1 Applicable Security Services

Operating system security services are specified in terms of controlling the access of subjects, or processes acting on their behalf, to data, functions, hardware, and software resources of a system. Access control policies include discretionary access control (DAC) and mandatory access control (MAC). In addition to access control, other security services include individual user identification and authentication (I&A), data confidentiality, integrity (e.g., system integrity, Trusted Computing Base (TCB) integrity, label integrity, data integrity), security audit, object reuse, and availability of service and data.

3.1.2 DODIIS Security Standards

The following documents have been selected as the DODIIS security standards for operating systems: *Department of Defense Trusted Computer System Evaluation Criteria* (TCSEC), DOD 5200.28-STD [9]; *Security Requirements for System High and Compartmented Mode Workstations* (CMWREQs), DDS-2600-5502-87 [10]; and the *Compartmented Mode Workstation Evaluation Criteria, Version 1 (Final)* (CMWEC), DDS-2600-6243-91 [11]. The IEEE *Portable Operating System for Computer Environments (POSIX) - Security Interface* (POSIX.6) [12], when released as an IEEE standard, will be adopted as a DODIIS standard. Additional standards that are being monitored for potential inclusion within the DODIIS Reference Model are the Federal Criteria standards currently being developed by NIST and the National Security Agency (NSA).

The TCSEC or "Orange Book," a DOD standard, is hierarchically ordered into a series of evaluation classes in which each class embodies an increasing degree of computer security protection and assurance. The CMWREQs identifies minimum security requirements for system high and compartmented mode workstations. The technical compartmented mode requirements defined within the CMWREQs have since been superseded by the CMWEC. The CMWEC provides an interpretation of the compartmented mode requirements defined within the CMWREQs in terms of the TCSEC. The CMWEC, along with the TCSEC, will form the basis upon which DIA and the National Computer Security Center (NCSC) will carry out future joint CMW product evaluations. The IEEE POSIX.6 standard will define a standard interface and environment for computer systems that require a secure environment and is intended for system implementors and application software developers. POSIX.6 defines five independent, optional security mechanisms that will become integrated into the POSIX System Application Program Interface (API) standard [13]. These mechanisms are ACLs, security auditing, least privilege, MAC, and information labeling.

NIST and NSA have entered into a joint project to develop a series of FIPS PUBS specifying new Federal Criteria (FC) for trusted systems. These FIPS PUBS are collectively being called the "Federal Criteria FIPS" (FC-FIPS). The first set of FIPS will be an enhancement and modernization of the current TCSEC coupled with an expansion of the existing product evaluation process. Version 1 of the FC, targeted for completion as a draft by fall 1992, is intended as an evolutionary modernization of the existing TCSEC. The *Minimum Security Functional Requirements for Multi-User Operating Systems* [14] will be used as input within the FC development process. Version 1 and its supporting documents will form the foundation for new system security criteria as a first step. DIA plans to include the FC-FIPS as DODIIS standards once they are published.

3.2 NETWORK SERVICES

Network services provide support to distributed applications requiring data access and applications interoperability in networked environments. Network services include data communications, transparent file access, distributed computing services, network management services, and network security services.

3.2.1 Applicable Security Services

The network security services are those defined in *ISO 7498-2-1988(E)* [15]. This document provides a general description of security services and related mechanisms, which may be provided by the OSI Basic Reference Model, and defines the positions within the OSI Basic Reference Model where the security services and mechanisms may be provided. The security services defined are authentication, access control, data confidentiality, data integrity, and non-repudiation. ISO 7498-2 also identifies pervasive security mechanisms that are not specific to any particular security service. These mechanisms include trusted functionality, security labels, event detection, security audit trail, and security recovery.

3.2.2 DODIIS Security Standards

The near-term DODIIS standards for secure networking are associated with the DOD internet protocol suite, whereas the long-term objective for DODIIS is to employ the security protocols defined within the Government OSI Profile (GOSIP). GOSIP is the U.S. Government specification for implementing international standards based on the seven-layer OSI model. The DOD internet protocol suite is the focus of the DRM since the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite is currently the installed base of the DODIIS community. The DODIIS community plans to transition to GOSIP as OSI-based COTS products become widely available. The following sections discuss the DODIIS standards and related developments for network security.

3.2.2.1 DNSIX

The near-term security requirements for DODIIS networking were defined by DIA in the 1986 publication *DODIIS Network Security Architecture and DODIIS Network Security for Information Exchange (DNSIX)* [16]. This document includes requirements based on DIA Manual (DIAM) 50-4, extensions to those requirements for networking (particularly for the DODIIS hosts), and additional network-specific requirements. Many of those security requirements will be met in DODIIS by software that will be added to the DODIIS hosts or to their network interface devices. This software is called DNSIX. DNSIX consists of several modules to allow for compartmented mode and system high hosts to exchange information over a network. Since this publication, additional documents have been developed that extend DNSIX for use in local area networks (LANs). These documents are the *DNSIX Interface Specifications, Version 2.1* [17] and the *DNSIX Detailed Design Specification, Version 2.1* [18].

3.2.2.2 Internet Engineering Task Force (IETF)

The IETF is a task force of the Internet Engineering Steering Group (IESG), a group that reports to the Internet Architecture Board (IAB). The IETF has a number of security-related working groups which are coordinated by the Security Area Advisory Group (SAAG). The working groups include the Commercial IP Security Option (CIPSO) working group, the Common Authentication Technology (CAT) working group, and the Privacy-Enhanced Electronic Mail (PEM) working group. Recent Request for Comments (RFCs) that have been published include RFC 1108 [19], RFC 1244 [20], and RFC 1281 [21]. Future Internet standards resulting from the IETF security working groups are being monitored for potential inclusion within the DRM.

3.2.2.3 Defense Message System (DMS)

The DMS has a target architecture that integrates the Automatic Digital Network (AUTODIN) and the Defense Data Network (DDN), uses International Telegraph and Telephone Consultative Committee (CCITT) X.400 message handling and CCITT X.500 directory services, and provides writer-to-reader security within the Secure Data Network System (SDNS) developed by NSA [22]. The DMS transition strategy is evolutionary and is in three phases, designed to transform DOD messaging from the 1989 baseline to the 2008 target architecture. DMS is transitioning to an architecture that employs GOSIP protocols and plans to use COTS products or products using non-developmental item (NDI) technology whenever possible. DIA is an active participant within DMS and plans to transition to the DMS target architecture for message handling to ensure interoperability.

3.2.2.4 GOSIP Security Protocols

The long-term security requirements for DODIIS networking will be provided by security protocols defined within GOSIP. In particular, security protocols developed within the SDNS program [23] are targeted for the DODIIS community. The SDNS program, a joint program of the U.S. Government and computer industry representatives, has developed OSI security protocol specifications for data networking. The SDNS protocols defined are the Security Protocol at Layer 3 (SP3), the Security Protocol at Layer 4 (SP4), the Message Security Protocol (MSP), and the Key Management Protocol (KMP) [24]. The SP3 and SP4 specifications are in the process of becoming ISO standards and are referred to as the Network Layer Security Protocol (NLSP) and Transport Layer Security Protocol (TLSP), respectively. MSP will be initially applied within DMS to provide message handling security services.

An additional standard being monitored includes the Standard for Interoperable LAN Security (SILS) (IEEE 802.10) [25]. SILS will provide a standard protocol for protecting LAN traffic and will also specify methods of key management and system/security management with supporting protocols.

3.2.2.5 DODIIS Evolutionary Path for Network Security

After considering the advances made by industry in developing secure networking standards, DIA believes that industry has taken initiatives to define and develop capabilities that exceed the DNSIX specifications. Consequently, DIA's policy with regard to the development of compartmented mode networking via DNSIX has changed, though interoperability with the present DNSIX 2.1 specifications will continue to be a minimum requirement. The compartmented mode networking policy is shifting to allow DIA and the DODIIS community to take advantage of the emerging technical security standards in industry. DIA and DODIIS members are participating in industry forums to ensure that the government's functional requirements for network security are included in emerging standards. Two prominent industry forums are the Trusted Systems Interoperability Group (TSIG) and the Project MAX consortium. TSIG and Project MAX memberships have substantial overlap, and Project MAX continues to be influenced by TSIG (and vice-versa). The TSIG is an industry-sponsored group working toward interoperability standards of CMWs and B-level trusted products. Project MAX was formed to fund an actual implementation of trusted networking technology, known as MaxSix. By encouraging industry in these efforts, the government should be able to purchase interoperable network security software as COTS products.

DODIIS will evolve over time to rely upon COTS products for network security. An evolutionary strategy has been developed to provide guidance to the community. The evolutionary path that DODIIS intends to pursue is listed below:

- Develop a baseline DNSIX, Version 2.1 through DMB action. This event occurred with the formal publication of the DNSIX Version 2.1 specifications, dated October 1991.
- Adopt MaxSix as DNSIX 3.0, provided suitable MaxSix interface specifications are available in the public domain and that MaxSix maintains interoperability with DNSIX Version 2.1.
- Work with the TSIG and Project MAX Consortium to develop and sponsor a "public domain" commercial set of trusted network security interface standards, with DNSIX functionality and interoperability as a required subset. The standards will include trusted sessions, Trusted Network File System (TNFS) [26], and a trusted implementation of the X Window System for user interface services.
- Move toward adopting the resulting standards as DNSIX 4.0 in 1993-1994 timeframe.
- Plan to transition to GOSIP security protocols as they become standards and are available as COTS products.

3.3 USER INTERFACE SERVICES

User interface services define how users interact with an application. User interface services include client-server operations, window object definition and management, presentation management, window management, dialogue support, and user interface security services.

3.3.1 Applicable Security Services

User interface security services include the definition and execution of types of user access to objects within the purview of user interface systems, such as access to windows, menus, icons, and the display of security labels.

3.3.2 DODIIS Security Standards

The *Department of Defense Intelligence Information System Systems Style Guide* [27] and the *Compartmented Mode Workstation Labeling: Source Code and User Interface Guidelines*, DDS-2600-6215-91 [28] are the DODIIS standards for user interface security services. These documents address the security portion of the user system interface (USI) and are intended for programmers who are developing applications for CMWs.

Appendix B of the DODIIS Style Guide seeks to provide a uniform and consistent USI across all CMW applications within the DODIIS community. DDS-2600-6215-91 describes a demonstration program that includes a user interface for setting and changing information labels, sensitivity labels, and clearances. Also included are DIA guidelines for implementing such user interfaces. Additional guidance on the usage of the DIA labeling software can be found in Appendix E of DDS-2600-6243-91 (the CMWEC).

3.4 OBJECT MANAGEMENT SERVICES

Object management services deal with the management of structured data, alphanumeric text, imagery, graphics, and other media, such as voice and video. Object management services include data dictionary/directory services, database management system (DBMS) services, data access services, distributed data services, and data management security services.

3.4.1 Applicable Security Services

Object management security services are not completely defined for all of the types of objects (e.g., imagery, graphics) within this service area. Secure voice capabilities and programs are available across the DOD [29]. The data management security services defined are those associated with a DBMS. Data management security services include access control (DAC, MAC, content-dependent and context-dependent access control), individual user I&A, data confidentiality, data integrity (e.g., domain integrity, entity integrity, referential integrity, label integrity), security audit, object reuse, and availability of service and data.

3.4.2 DODIIS Security Standards

Very few standards address data management security services, although standards bodies are in the process of developing standards. These standards bodies include ANSI X3H2 and ANSI X3H7, both technical committees of ANSI SPARC. ANSI X3H2 is active with developing Structured Query Language (SQL) security extensions whereas X3H7 (formerly the Object-Oriented Data Base Task Group), plans to address security within Object Data Management. The Database Management Security and Privacy Task Group (DMSPTG), an ANSI SPARC study group, is developing a reference model that will provide a framework for security/privacy for data management [30].

The NCSC *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria* (TDI) [31], released in April 1991, extends the evaluation classes of the TCSEC to trusted applications in general, and to DBMSs in particular. The TDI provides guidance for the design, implementation, and evaluation of trusted DBMSs. The TDI should be regarded as an interim standard since this NCSC publication interprets the TCSEC for database systems. In addition, several trusted DBMS products are being evaluated against this TCSEC interpretation.

3.5 OBJECT INTERCHANGE SERVICES

Object interchange services provide specialized support for the interchange of text, imagery, graphics, and multimedia between applications (either on the same or different platforms) primarily through the specification of encodings or data structures. Object interchange services include product data interchange services and object interchange security services.

3.5.1 Applicable Security Services

Object interchange security services are used to verify and validate the integrity of specific types of data interchange. Examples of such security services include non-repudiation, authentication, data integrity, and access control. Object interchange security services will depend on network security services.

3.5.2 DODIIS Security Standards

The DODIIS standards defined for object interchange security services are closely coupled with those defined within network security services. Future security standards to be adopted for this service area include the NIST Digital Signature Standard (DSS) [32] [33], the Open Document Architecture and Interchange Format (ODA/ODIF) security extensions, and the CCITT Electronic Data Interchange (EDI) security extensions. The NIST DSS, currently under review as a proposed FIPS, will specify a digital signature algorithm that can be used to generate a digital signature. Digital signatures are used to detect unauthorized modifications to data and to authenticate the identity of the user who generates the signature. The purpose of the ODA/ODIF standard is to facilitate the interchange of documents. Both the European Computer Manufacturers Association (ECMA) and the ISO are developing ODA security extensions [34]. Security is also an active issue in several other specific OSI applications such as the EDI extensions to CCITT X.400 [35].

Additional standards address security formats. DODIIS standards include the DNSIX Network Audit Trail (NAT) format, the DNSIX Network Level Module (NLM) label format, and the CMW label encodings format as defined in DDS-2600-6216-91) [36]. Future standards to be adopted include the POSIX.6 audit trail record format [12] and the GOSIP security label format [37] [38].

3.6 SYSTEM MANAGEMENT SERVICES

System management services are required in the DRM to provide a means for monitoring and controlling the various components of the information system architecture. OSI system management standards have been partitioned into five Specific Management Functional Areas (SMFAs). The five SMFAs are configuration management, fault management, performance management, accounting management, and security management. These SMFAs have overlapping requirements, and management information and functions applicable to one SMFA are often applicable to another SMFA. Security services and standards are those associated with security management.

3.6.1 Applicable Security Services

Security management, an integral part of system management, is concerned with the management of security services and mechanisms. Included within security management is overall security policy management. A security policy is enforced across all of the components of the DRM. Security management services include system security management, security service management, and security mechanism management. Security mechanism management functions include maintaining sensitivity labels (e.g., operating system, router), managing user clearance levels, performing audit administration, maintaining security databases (e.g., password files, MaxSix databases), and distributing cryptographic keys..

3.6.2 DODIIS Security Standards

DODIIS standards for security management include the security services provided by SNMP and the services that will be defined in the Government Network Management Profile (GNMP) [39]. SNMP security extensions have been defined by the IETF. The GNMP will be the standard reference for all Federal Government agencies to use when acquiring network management functions and services. With respect to security, the GNMP will identify security options, management security services, and management security standards activities.

The audit trail function is an element of security management. As a result, the DNSIX network audit trail is included as a standard in this service area. An Internet draft under consideration is the Security Information Transfer Protocol (SITP) [40]. SITP is being proposed as a standard for the transfer of audit data across TCP/IP networks.

Additional standards being monitored for potential inclusion within the DRM are those associated with the work items of ISO SC21 Working Group #4. This particular working group is developing standards in the areas of alarm reporting, security alarm reporting, and the security audit trail function [41].

4.0 SUMMARY

The DODIIS community is striving toward a uniform standards-based architecture. The DRM provides a foundation for the development of architectures and implementation/transition plans, and the selection of products that will meet the goals and objectives of the DODIIS community. When applied, the DRM will serve to maximize interoperability and portability among DODIIS programs and applications, and reduce costs through the use of common services. The DRM is not a specific system design, but a set of standards and guidelines that can be applied to DODIIS programs and site architectures. The DRM is also evolutionary, since not all of the needed standards (e.g., security standards) have emerged.

Security is an essential element of the DRM. While DODIIS recognizes that security is integrated within each of the DRM components, security requirements and standards for several of the components (e.g., Object Management Services) have not been formally defined. The DRM security enhancements and revisions are targeted for FY92. This paper provided a snapshot of this activity, especially with the identification of the applicable security services and associated security standards for specific service areas of the DRM. The DRM is expected to continue to provide guidance and direction to DODIIS planners and engineers for the development of site security architectures.

ACKNOWLEDGMENTS

The author wishes to thank Walt Jablonski, DIA/DS-SIM, for his direction and encouragement during the development of this paper. The author also thanks Robert Brown, Les Fraim, Tom Gregg, Patty Guay, Gary Huber, Dr. Albert Jeng, Carol Oakes, Mindy Rudell, and Lorrayne Schaefer of The MITRE Corporation for their helpful comments and insights on the technical material in prior drafts of this paper.

REFERENCES

1. DIA, *A Framework for Evolution of the Department of Defense Intelligence Information System (DODIIS)*, July 1991.

2. Joint Chiefs of Staff (JCS), *Department of Defense Dictionary of Military and Associated Terms*, Joint Publication 102 (Formerly JCS PUB 1), 1 December 1989.

3. DIA, *DODIIS Reference Model for the 1990s*, 18 October 1991.

4. NIST, *Application Portability Profile (APP), The U.S. Government's Open System Environment Profile, OSE/1 Version 1.0*, NIST Special Publication 500-187, April 1991.

5. ISO, *Information Processing Systems - Open Systems Interconnection - Basic Reference Model*, International Standard 7498, 1984.

6. DISA, Center for Information Management, *Technical Reference Model for Information Management*, Version 1.2, 15 May 1992.

7. IEEE, *Draft Guide to the POSIX Open Systems Environment*, P1003.0 Draft 14, November 1991.

8. DIA, *DODIIS Documentation Management Plan*, 21 November 1991.

9. DOD, *Department of Defense Trusted Computer System Evaluation Criteria* ["Orange Book"], DOD 5200.28-STD, December 1985.

10. DIA, *Security Requirements for System High and Compartmented Mode Workstations*, DDS-2600-5502-87, November 1987.

11. DIA, *Compartmented Mode Workstation Evaluation Criteria, Version 1 (Final)*, DDS-2600-6243-91, November 1991.

12. IEEE, *Draft Standard for Information Technology - Portable Operating System Interface (POSIX) - Security Interface*, P1003.6 Draft 12, ISO/IEC JTC 1/SC22/WG15 N046R1, September 1991.

13. ISO, *Information Technology - Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API)*, ISO/IEC 9945-1, 1990.

14. NIST, *Minimum Security Functional Requirements for Multi-User Operating Systems*, Issue 1, 27 January 1992.

15. ISO, *Information Processing Systems- Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture*, International Standard 7498-2-1988(E), 1988.

16. DIA, *DODIIS Network Security and DODIIS Network Security for Information Exchange (DNSIX)*, DRS-2600-5466-86, 5 May 1986.

17. DIA, *DNSIX Interface Specifications, Version 2.1 (Final)*, DDS-2600-5984-91, October 1991.

18. DIA, *DNSIX Detailed Design Specification, Version 2.1 (Final)*, DDS-2600-5985-91, October 1991.

19. Kent, S., *U.S. Department of Defense Security Options for the Internet*, RFC 1108, November 1991.

20. Holbrook, J. P., and J. K. Reynolds, *Site Security Handbook*, RFC 1244, July 1991.

21. Pethia, R., S. Crocker, and B. Fraser, *Guidelines for the Secure Operation of the Internet*, RFC 1281, November 1991.

22. Shirey, R. W., "The Defense Message System," *Computer Communications Review*, Volume 20, Number 5, October 1990.

23. Karp, B. C., L. K. Barker, and L. D. Nelson, "The Secure Data Network System," *AT&T Technical Journal*, May/June 1988.

24. NIST, *Secure Data Network System (SDNS) Network, Transport, and Message Security Protocols*, NISTIR 90-4250; *Secure Data Network System (SDNS) Access Control Documents*, NISTIR 90-4259; *Secure Data Network System (SDNS) Key Management Documents*, NISTIR 90-4262, February 1990.

25. Barker, L. K., and K. Kirkpatrick, "The SILS Model For LAN Security," *Proceedings of the 12th National Computer Security Conference*, NIST/NCSC, October 1989.

26. Glover, F., *Request for Comments on a Specification of Trusted NFS (TNFS) Protocol Extensions*, Internet Draft, 24 May 1992.

27. DIA, *Department of Defense Intelligence Information Systems Style Guide*, October 1991.

28. DIA, *Compartmented Mode Workstation Labeling: Source Code and User Interface Guidelines, Version 1 (Final)*, DDS-2600-6215-91, November 1991.

29. Defense Communications Agency, *Department of Defense Secure Voice Architecture*, 18 May 1990.

30. Ashby, V., and L. Schlipper, "Security Standards for Object Data Management Systems," *Computer Standards & Interfaces 13*, Elsevier Science Publishers, 1991.

31. NCSC, *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria* [the "TDI"], NCSC-TG-021, Version-1, April 1991.

32. NIST, *A Proposed Federal Information Processing Standard for Digital Signature Standard (DSS)*, Technical Report FIPS PUB XX, Draft, August 1991.

33. Communications of the ACM, "The Digital Signature Standard Proposed by NIST," *Communications of the ACM*, Vol. 35, No. 7, July 1992.

34. Murphy, K. V., and G. Soberg, "ODA and POSIX: Label Liaisons," *Proceedings of the Sixth Annual Computer Security Applications Conference*, IEEE Computer Society Press, December 1990.

35. Ford, W., "International Activities ISO/IEC and CCITT," *Proceedings of the Sixth Annual Computer Security Applications Conference*, IEEE Computer Society Press, December 1990.

36. DIA, *Compartmented Mode Workstation Labeling: Encodings Format (Final)*, DDS-2600-6216-91, November 1991.

37. NIST, *Standard Security Label for the Government Open Systems Interconnection Profile*, Draft FIPS PUB XXX, 3 March 1992.

38. NIST, *GOSIP Chapter 6 Security Options (Proposed Draft for Version 3)*, March 1992.

39. NIST, *Proposed Government Network Management Profile (GNMP)*, Version 1.0, 8 March 1991.

40. Dempsey, J., C. Feil., and N. Lewis, *Security Information Transfer Protocol*, Internet Draft, April 1992.

41. Ward, R., and P. Skeffington, "Network Management Security," *Proceedings of the Sixth Annual Computer Security Applications Conference*, IEEE Computer Society Press, December 1990.

SEPARATION MACHINES

Jon Graff

Amdahl Corporation
1250 Arques Avenue (M/S 119)
P.O. Box 3470
Sunnyvale, CA 94088-3470

Summary

This paper consists of 3 sections:

- An Introduction which defines the term "Separation Machine".
- A description of Separation Machine Security labels as an example of the changes required in thinking about security based on the Orange and Red Books.
- Suggested applications of the separation machine concept to "real" security problems.

Introduction

A Separation Machine is a computer that is divided (multiplexed) into separate environments (called "Domains") for independently running "System Control Programs" (SCPs). Separation Machines have been characterized as "Virtual Machine Monitors" (VMMs) [1,2,3]. In a Separation Machine, the SCPs run directly on the Separation Machine's hardware and it appears to each SCP that the SCP has sole possession of a complete and separate machine and that machine's facilities. The operating systems within the Separation Machine's Domains are examples of SCPs.

Separation Machines are a special type of "non-interference" machines. They are based on the Rushby "Separation" Security Policy Model [4][1] and provide a great deal of inherent security for each of its Domains [5]. In a Separation Machine, the isolation security policy is manifested in the fundamental architecture of the machine. The Domains co-exist on the computer under the supervision of the Separation Machine. The Separation Machine enforces the separation of each of the Domains by giving each Domain a unique time slice of the CPU as well as assigning each Domain its own disjoint set of resources such as storage and channels. During its time slice, the Domain and its SCP have exclusive use of the computer facilities and the Domain's resources (CPU, storage and channels). Additionally, once the Separation Machine assigns a resource to a Domain, that Domain maintains sole and exclusive possession and access to that resource until the Separation Machine oversees that resource's release.

© Copyright 1992 Amdahl Corporation

---

[1] In personal communications, Ron Watro, MITRE Corp., Bedford, MA, has indicated two ambiguities in Rushby's presentation. Dr. Watro is currently writing a clarification of the issues. The issues are:

- o in Theorem 1, a false assumption is made about transitivity (this was first pointed out by Kelem and Fiertag [2]).
- o in the generalized proof, the six separation conditions are not shown to imply secure isolation.

When the Domain's time slice expires, the Separation Machine puts the SCP and its Domain into a state of "suspended animation." At the beginning of a time slice, the Separation Machine reactivates the Domain and its SCP into exactly the same active state the Domain and the SCP were in immediately prior to being placed into suspended animation.

At the time of Domain creation, the System Security Authority (SSA) supervising the Separation Machine defines the environment for the SCP. While running in its Domain, the SCP knows only the facilities that the Separation Machine provides it. The SCPs have no discretion on what facilities to which they have access, but of course, each SCP has discretion as to which facilities it wishes to use out of those available to it.

Thus, the Separation Machine enforces on the SCPs a very strict Mandatory Access Control (MAC) policy. This MAC policy is based on a security policy enforced at the will of some privileged user(s) (e.g., the SSA), and not at the will of non-privileged users. This is in contrast to MAC based on sensitivity labels as defined in the "Orange" Book [6][2].

A Separation Machine may be thought of as a bank vault that contains a set of safe-deposit boxes. Each SCP resides within its own safe-deposit box, i.e., its own Domain. Not only are the SCPs safe from unauthorized external influences, but more importantly, each SCP is separated from any influence on each other, i.e., the contents of one safe deposit box cannot know about or influence the contents of another safe deposit box. Indeed, a Separation Machine provides such robust security that the SCPs do not have to supplement their individual security policies in order to maintain their security assurances.

In Figure 1, the Separation Machine is schematically represented by the "brick wall" and the "windows" are the Domains. The figure schematically shows the Separation Machine fully populated by seven independently operating SCPs.

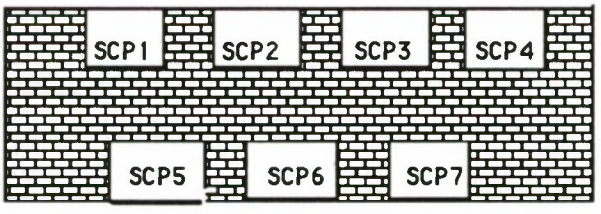


Figure 1: A Separation Machine showing "populated" Domains

Security Labels in Separation Machines

The security policies set forth in the Orange Book [6] and Red Book [7] are based on the trusted "reference monitor" concept with an emphasis on the Bell and LaPadula Security Policy Model [8]. The reference monitor's function is to ensure that access is only permitted when the subject's and object's labels meet the requirements set forth in the security policy. The

---

[2] p. 114: "**Mandatory Access Control** - A means of restricting access to objects based on the *sensativity* (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity." (emphasis added)

concept calls for the trusted reference monitor to examine "labels" to determine if a "subject" (an active agent, such as a calling program or a human) has permission to access an "object" (passive resource such as a piece of data). The subject's label indicates the characteristics that an object must have in order for a subject to be permitted to access the object. The object's label identifies what characteristics the subject must have in order to be permitted to access the object.

The Bell and LaPadula Family of Security Policies

The Bell and LaPadula family of security policies (BLFSP) are based on a reference monitor that requires sensitivity-levels as a mechanism for policy enforcement. The model is based on the environment in which multiple subjects may have access to multiple objects. In the BLFSP the reference monitor adjudicates the access control between subjects and objects by comparing their "sensitivity labels" according to the Mandatory Access Control (MAC) policy. The reference monitor permits a subject to access an object only if the subject's and object's sensitivity labels fulfill the requirements of the security policy.

The important point of this discussion is that sensitivity labeling is a required part of maintaining the BLFSP.

The Rushby Separation Security Policy

In contrast to the more familiar BLFSP, the Rushby Separation Security Policy is "Isolation" enforced by the mechanism of "Separation." The Rushby Isolation Policy requires the segregation of each subject and its objects from any other subject and their objects. Therefore, it is the Separation Machine's responsibility to ensure that each Domain, and therefore its respective SCP, is kept totally separate and without knowledge of or access to any other Domain's resources.

Comparison of the BLFSP and the Rushby Policy

The Rushby policy of Isolation as implemented in the separation security model does not have the same requirements as the BLFSP. Both obtain Mandatory Access Control (MAC) but through different mechanisms. A Separation Machine ensures MAC by total separation, i.e., the Separation Machine's MAC is Separation; in contrast, the BLFSP's MAC requires the adjudication of the sensitivity labels of the subjects and the objects.

Table I shows the two types of Security Labels. The BLFSP requires Sensitivity labels, whereas the Isolation policy requires "Separation" labels. Each SCP within the Separation Machine defines its own individual security policy. An SCP may base its security policy on any one of a number of different security policies; one choice being those belonging to the BLFSP. Therefore the individual SCPs may, indeed, require sensitivity labeling. However, it is very important to note that the operations within the SCP are out of the purview and responsibility of the Separation Machine.

Table 1: Types of Security Labels		
Major Class of Labels	**Sub-Labels**	**Function**
Sensitivity		Used by the BLFSP Reference Monitor to determine if a subject should have access to an object.
	hierarchical	Indicate "classification level," e.g., Top Secret, Secret, or Confidential. These labels correspond to the security risk of having the information compromised.
	non-hierarchical	Indicate "compartments." Compartments are subgroups of a classification level (e.g., artillery, armor, infantry or Army, Navy and Air Force). These labels do NOT exist independently of the classification level.
Separation		Used by the Separation Machine reference monitor to determine if a Domain has possession of a resource.

"Sensitivity" Labeling contrasted with "Separation" Labeling

Traditional "Sensitivity labels" supporting the reference monitor model

The BLFSP require sensitivity labeling. Sensitivity labeling has two aspects, a hierarchical part and, if required, a subservient, non-hierarchical part. The hierarchical part of the label refers to the classification level or "security sensitivity," e.g., Top Secret, Secret, and Confidential. The hierarchical classification levels define the security risk of the unauthorized release of the information. Within each classification level there may exist "compartments" that define areas of "the need to know" or access requirements. These compartments are "non-hierarchical" because they require the same clearance for access, however they each have a different "need-to-know" requirement.

In the BLFSP, each subject is assigned permission to access information or perform tasks based on the subject's security risk (classification level) and "need to know" (compartment). The same labeling is applied to objects. These sensitivity labels must be protected from unauthorized changes and therefore strict requirements are made on how the sensitivity labels are generated, used, monitored and protected. A MAC policy mandates the labels assigned in an automatic, prescribed manner.

The Separation Machine "labeling" solution[3]

The Separation Machine does not have or need the traditional "sensitivity labels" to ensure MAC. The Separation Machine ensures MAC by enforcing the strict separation of the Domains through hardware. This separation begins at the time of Domain activation. At that time, the

[3]Parts of this section have been presented at the Standard Security Label for GOSIP, An Invitational Workshop, June, 1991.

SSA assigns specific resources that the Domain may use. When the Domain receives a resource, the Separation Machine assigns the Domain's identity to that resource.

In Rushby's discussion each Domain has a different "colour" that is used to assist in separation. Thus the "colours" are exactly equivalent to the Separation Machine Domain identifiers and may be thought of as "Separation" labels. A "snapshot" of a Separation Machine shows that each interaction between subjects (SCPs and Domains) and objects (e.g., storage, and channels) is, in fact, based on separation labels: every external storage device is attached to one or more channels of the same "colour." Thus each of these channels is attached to exactly one Domain of the same "colour." All storage has the same "colour."

It must be emphasized that the Separation Machine does not need or require sensitivity labels to enforce the MAC through the security policy of Separation. In the Separation Machine, MAC is maintained with separation labels. The separation labels permit the Separation Machine to ensure that the Domains are totally separate and independent. The operation of the individual SCPs within the Domains are of no concern to the Separation Machine. Each SCP will have its own security policy and these SCP security policies (e.g., MAC policies) may require the traditional sensitivity labels.

Examples of other "Security" Labels

Table 2 presents other examples of different types of security labels required by various different security policies.

Table 2: Various Security Policies and Their Mechanisms and Security Labels

Policy	Mechanism	Security Label
Bell - LaPadula	Access Control Monitor	Sensitivity
Biba	Integrity Control Monitor	Integrity
Clark - Wilson	Integrity Control Monitor	Integrity
Chinese Wall	Individual Responsibility for Separation	Identity
Rushby	Separation	Separation

Suggested Security Uses for Separation Machines

As shown in Figure 1, the Separation Machine permits the consolidation of several independent and separate SCPs each running on its own computer systems to run on a single Separation Machine computer system. The separation in a Separation Machine is so strong that:

- each SCP may have its own security policy so, for example, it is possible to implement Bell and LaPadula (B-P)[8], Biba [9], Clark/Wilson [10], Rushby or Chinese Wall [11] security policies in different SCPs that are simultaneously running in one Separation Machine, see Figure 2.

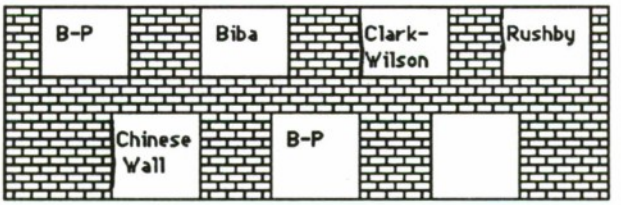


Figure 2: A Separation Machine, simultaneously implementing Different Security Policies each in Different Domains

- depending on the security assurance of the Separation Machine, it is possible to run SCPs having different security assurance levels within the same Separation Machine, see Figure 3. Thus in a Separation Machine having a security assurance of A1, it would be possible to run an SCP with a D rating in one domain and an SCP with an A1 in a second domain without diminishing the A1 SCP's security assurance.

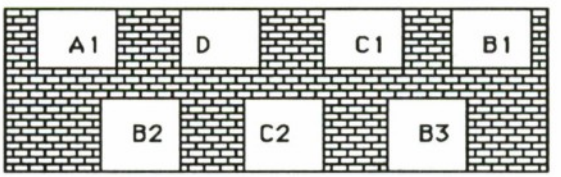


Figure 3: A Separation Machine Accredited to A1 simultaneously running Different Security Levels each in Different Domains

As based on the "Yellow Book" [12], the security assurance ascribed to the Separation Machine defines the possible usage of the Separation Machine. The higher the security assurance, the more complex the mixture of sensitivity levels that the machine can accommodate. For example, if the Separation Machine has the security assurance of B1, it may be used in two possible ways, as shown in Figures 4 and 5.

- Based on the Yellow Book, the Separation Machine may be used to accommodate a mixture of SCPs that have a security difference of 1. Thus the SCP's may have a rating difference of Unclassified (U) and Confidential (C) or Confidential (C) and Secret (S) as shown in Figure 4.

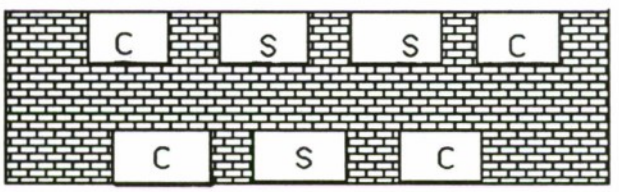


Figure 4: A Separation Machine Accredited to B1 running Domains containing SCPs having two security levels.

- Alternatively, the Separation Machine may run SCPs at the Top Secret (TS) level that have up to two different compartments (TS_a and TS_b) as shown in Figure 5.

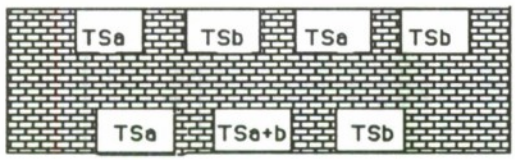


Figure 5. A Separation Machine Accredited to B1 running Domains containing Top Secret (TS) SCPs having two different compartments (a and b).

If the Separation Machine is assured to the B2 level, the Separation Machine may be used as schematically shown in Figures 6 and 7.

- In Figure 6, the Separation Machine can contain SCPs running at a security separation of 2; thus the Separation Machine can contain both Secret (S) and Top Secret (TS) SCPs. Similarly, but not shown, a Separation Machine could simultaneously run Unclassified (U), Confidential (C) and Secret (S) SCPs.

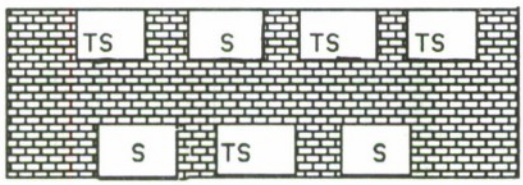


Figure 6: A Separation Machine Accredited to B2 running Domains containing Secret (S) and Top Secret (TS) SCPs.

- As shown in Figure 7, the Separation Machine can contain SCPs running at Top Secret that have an unlimited number of compartments.

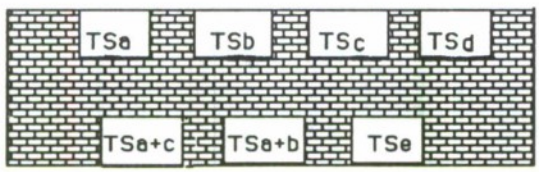


Figure 7: A Separation Machine Accredited to B2 running Domains containing Top Secret SCPs with several different compartments.

In addition to permitting the consolidation of separate computer systems onto a single Separation Machine, the Separation Machine presents a solution to the "information downgrade" problem, as illustrated in Figures 8, 9 and 10. The "information downgrade" problem, in the context of this paper, is the problem that occurs when SCPs of different security sensitivity levels need to communicate. According to the BLFSP, the lower security level SCP may only write information "up" to the higher security level SCP and the upper security level SCP may only read information that resides "down" in the lower security level SCP. Therefore, the upper security level SCP cannot even send message acknowledgments (e.g., ACKs and NACKs) to the lower SCP to complete a communication loop. The problem is more acute than simply

fulfilling a message protocol. There are many instances where high security level information in one SCP must be rapidly "downgraded" and sent to a lower level security SCP. For instance, as a hypothetical example, selections from the Joint Chief of Staff's target list must be sent to the individual mission commanders through a series of intermediate and successively lower security levels.

- Figure 8 shows an example of the most common current solution. Information from a Top Secret SCP on one machine must be transferred to a removable medium such as a print-out or a disk and then manually passed to a human intermediate who examines the contents and approves the transfer. Only then may the information be placed on the Secret SCP running on the second machine. This physical transfer is referred to as "sneakernet" [13].

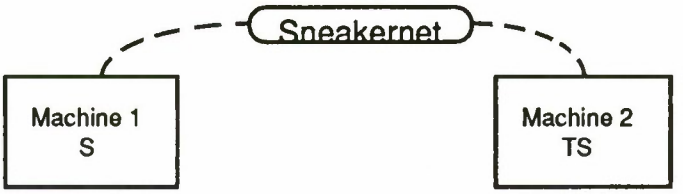


Figure 8: Information Downgrade using "Sneakernet" between two separate machines.

- Figure 9 presents a more automated approach. The TS and S SCPs, each residing in its own separate and distinct Domain, are connected by dedicated, defined channels through a trusted automated piece of hardware that is generically called a "Guard" (for examples see [14] and [15]). The trusted Guard contains trusted software that examines the information to be transferred and permits only predefined, acceptable data to pass between the SCPs.

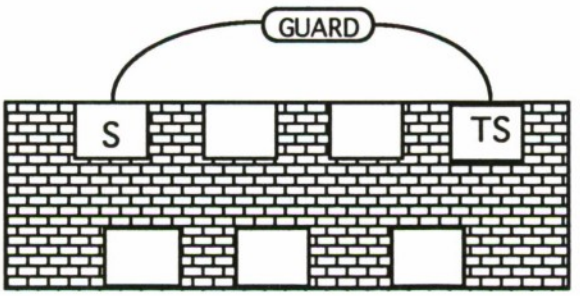


Figure 9: Information Downgrade in a Separation Machine using an External Guard

- Figure 10 presents a potential further enhancement. The TS and the S SCPs, each residing in its own separate and distinct Domain, are again connected by dedicated, defined channels through a trusted Guard. However, in this case the Guard resides within its own Domain in the Separation Machine. Thus the Separation Machine itself ensures that the only connection between the TS and S SCPs is possible through the Guard, and the Guard cannot be bypassed. The Guard is an SCP residing within a Domain provided by the trusted hardware of the Separation Machine. Consequently, only the Guard's software has to undergo the accreditation process; thus eliminating the need to design, build, evaluate and accredit the Guard hardware.

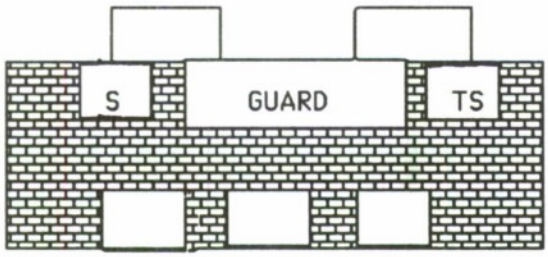


Figure 10: Information Downgrade in a Separation Machine using an Internal Guard

Conclusions

In conclusion, it has been shown that:

- Separation Machines are a special type of "non-interference" machines. They are based on the Rushby "Separation" Security Policy Model and provide a great deal of inherent security for each of its Domains.
- The Rushby Separation Security Policy requires a re-thinking of Orange/Red Book requirements such as Sensitivity Labels.
- The Separation Machine presents solutions to a number of security problems, and of particular importance, information downgrade.

References

[1] Paul A. Karger, Mary Ellen Zurko, Douglas W. Bonin, Andrew H. Mason, and Clifford E. Kahn, "A VMM Security Kernal for the VAX Architecture", Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy, May 7-9, 1990, Oakland, CA, pp. 2-19.

[2] Nancy L. Kelem and Richard J. Feiertag, "A Separation Model for Virtual Machine Monitors", Proceedings: IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, May 20-22, 1991, pp. 78-86.

[3] Russel, T. T. and Schaefer, M. Toward a High B Level Security Architecture for the IBM ES/3090 Processor Resource/Systems Manager™ (PR/SM™); 12th NCSC, October 1989, pp. 184-196.

[4] Rushby, J. M., "Proof of Separability", A Verification Technique for a Class of Security Kernal (Revised Version of SSM/11) Reprint (slightly expanded) of a paper presented at the 5th International Symposium of Programming, Turin, Italy, April 4-6, 1982. (Springer-Verlag LNCS No. 137 pp. 352-367).

[5] Robert W. Doran, "Amdahl Multiple-Domain Architecture", Computer (IEEE), Vol. 21 (10), October 1988, pp. 20-28.

[6] Department of Defense Standard: Department of Defense Trusted Computer System Evaluation Criteria, DOD 5200.28.STD, December 1985.

[7] National Computer Security Center Trusted Network Interpretation, NCSC-TG-005 Version-1, July 31, 1987.

[8] D. E. Bell and L. LaPadula, "Secure Computer System: Unified Exposition and Multics Interpretation", MTR-2997 Rev.1, MITRE Corp., Bedford, MA, 1976.

[9] K. J. Biba, "Integrity Considerations for Secure Computer Systems", National Technical Information Service, Springfield, VA, NTIS AD-A039324.

[10] David D. Clark and David R. Wilson, "A Comparison of Commercial and Military Computer Security Policies", Proceedings of the 1987 IEEE Symposium on Security and Privacy, April 27-29, 1987, Oakland, CA, pp. 184-194.

[11] Dr. David F.C.Brewer and Dr. Michael J. Nash, "The Chinese Wall Security Policy", 1989 IEEE Computer Society Symposium on Security and Privacy, May 1-3, 1989, Oakland, CA, pp. 206-214.

[12] Technical Rationale behind CSC-STD-003-85: Computer Security Requirements: Guidance for applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments, CSC-STD-004-85, June 25, 1985.

[13] James B. Hiller, "Sneakernet: Getting a Grip on the World's Largest Network, Proceedings of the 14th National Computer Security Conference, October 1-4, 1992, Washington, D.C., pp. 533-542.

[14] Michelle J. Gosselin, "The Development of a Low-to-High Guard", Proceedings of the 14th National Computer Security Conference, October 1-4, 1991, Washington, D.C., pp. 157-166.

[15] Michael F. Thompson, Roger R. Schell, Albert Tao and Timothy E. Levin, "Introduction to the Gemini Trusted Processor", Proceedings of the 13th National Computer Security Conference, October 1-4, 1990, Washington, D.C., pp. 211-217.

Software Forensics: Can We Track Code to its Authors?

Eugene H. Spafford
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907–1398
spaf@cs.purdue.edu

Stephen A. Weeber
Lawrence Livermore National Laboratory
P.O. Box 808, L–419
Livermore, CA 94550
weeber@llnl.gov

June 29, 1992

Abstract

Viruses, worms, trojan horses, and crackers all exist and threaten the security of our computer systems. Often, we are aware of an intrusion only after it has occurred. On some occasions, we may have a fragment of code left behind — used by an adversary to gain access or damage the system. A natural question to ask is "Can we use this remnant of code to identify the culprit?"

In this paper, we detail some of the features of code remnants that might be analyzed and then used to identify their authors. We further outline some of the difficulties involved in tracing an intruder by analyzing code. We conclude by discussing some future work that needs to be done before this approach can be properly evaluated. We refer to our process as *software forensics*, similar to medical forensics: we are examining the remains to obtain evidence about the factors involved.

Keywords: analysis, reverse engineering, evidence

1 Introduction

An aspect of both computer crime and computer vandalism that makes them more attractive activities is their anonymity. Whether the method of attack is virus, worm, logic bomb, or account breaking, tracing the actions back to an individual is generally an extremely difficult task. In one well-known case, Cliff Stoll's German hacker did not fear discipline even after being detected, trusting in the inability of anyone to trace his many network hops.[6] Authors of viruses distribute them without worry of being identified as the source. Participants in discussions in electronic forums tend to become more hostile than they ever would in face-to-face conversations; the anonymity of the transaction lowers their inhibitions.

Taking steps to remove the anonymity in computer use, such as more complete session logging and improved network protocols that include authentication information, can only help to discourage an attacker. However, there are limits to the strength of methods that can be economically employed. Furthermore, no method is 100% effective under all circumstances.

Often, the evidence remaining after a computer attack has occurred includes the instructions introduced into the system to cause the damage. Viruses, for example, usually leave their code in the infected programs. These remnants of an attack may take many forms, including programming language source files, object files, executable code, shell scripts, changes made to existing programs, or even a text file written by the attacker. It would be useful if these pieces of information could be utilized in a way that could help identify or confirm the source of the attack. This would be similar to the use of handwriting analysis by law

©1992 E. H. Spafford and S. A. Weeber

enforcement officials to identify the authors of documents involved in crimes, or to provide confirmation of the role of a suspect.

Handwriting analysis involves identifying physical features of the writing in question. A feature of the writing can be anything identifiable about the writing, such as the way the i's are dotted or average height of the characters. The features useful in handwriting analysis are the writer-specific features. A feature is said to be writer-specific if it shows only small variations in the writings of an individual and large variations over the writings of different authors.

Features considered in handwriting analysis today include shape of dots, proportions of lengths, shape of loops, horizontal and vertical expansion of writing, slant, regularity, and fluency.[12] The majority of features in most handwriting are ordinary. However, most writing will also contain features that set it apart from the samples of other authors, features that to some degree are unusual.[9] A sample that contains i's dotted with a single dot probably will not yield much information from that feature. However, if the i's are instead dotted with hearts, that feature may identify the author.

Identification of writer-specific features generally requires many samples. A person's handwriting is expected to change both as time passes and under different writing conditions. Too few samples can lead to misidentification of writer-specific features. Additionally, care must be taken in selecting samples that show the natural writing of an individual. Individuals often attempt to hide their identity by disguising their writing.

Identification of computer code by matching identifying characteristics should likewise be possible. Although programs do not exhibit the same types of physical features that handwriting does, programming, especially in a language rich in data types and control structures, has considerable room for variation and innovation. Even if coding is from detailed specifications, room exists for personalization. Programmers generally acknowledge that they each have a unique coding style. Using appropriate stylistic elements may help in the production, reuse, and debugging of code. Many texts recommend elements of style to use when programming, and often programmers integrate selected elements of others' styles into their own repertoire as they gain experience programming (cf. [3, 11, 13, 8, 21]).

Another method of verifying authorship is through the use of statistical inference. This technique has long been used to verify the authorship of prose, but its fundamental concepts appear directly applicable to software. Additionally, it overcomes one of the main criticisms of the techniques of handwriting analysis: subjectivity. Handwriting analysis depends significantly upon judgments made by the analyst. The techniques used in prose classification are rooted in statistical theory and have less opportunity for bias from the investigator.

Prose classification is generally conducted by identifying a large set of easily recognized features of the text. Often analyzed features include word and sentence length, the number of uses of selected words, percentages of nouns and adjectives, word pairs, and the use of comparatives and superlatives. Statistical theory is applied to the features found in works of both known and questioned authorship to produce a probabilistic determination of authorship. The advantage of this method lies in the fact that, while a single feature of a document is unlikely to provide much information, a large set of variables, such as the usage rates for dozens or hundreds of different words, stand a far better chance of indicating classification of the document.[15]

Such analysis should also prove useful in software forensics. Selecting several variables that may be measured from source code, a profile of an author's normal coding practices could be established from code of undisputed authorship. Then, using an established classification model, the unknown code may be compared.

Possible features of software that could prove differentiating under this technique include most of the features and metrics mentioned later in this paper. Program size, function size, Hamming distance between variable names, and ratio of comments to code could be easily measured. Just as some authors of prose show preferences for some words over others, programmers often demonstrate preferences for certain language constructs over others. This can be especially evident in feature-rich languages, such as Ada. These features might include a preference for `while` loops over `for` or `repeat..until` loops, use of function pointers, function overloading, and use of inheritance and polymorphism. Use of certain programming techniques

may also discriminate among authors. An author may favor certain data structures and system calls over others. For example, an author might show a preference for using hashed data structures over tree based structures.

A source of variables for analysis that bears further investigation is source code style analysis programs. `cstyle`, one such program, rates the programming style present in C programs, assigning numeric scores in eleven stylistic areas, including characters per line, percentages of comment lines, indented lines, and blank lines, space characters per line, number of different reserved words, `goto` use, number of included files, and use of constants.[1] It seems likely that given a large enough sample of source code, a stylistic profile could be developed for a programmer using this tool.

The keys to identifying the author of suspect code are selection of an appropriate body of code and identification of appropriate features for comparison. This may not be easy to do if the programmer has attempted to hide his authorship, or if appropriate sample code is not available. Nonetheless, our personal experience is such that we believe important features might still be present for analysis, in some cases. At the least, analysis of the characteristics of the code might well lead to the identification of suspects to examine further.

Additionally, if sufficient background research is done to establish a good statistical base, and if large enough samples of code are present, known statistical methods currently applied to determine authorship of prose may also be applied to code. These methods, although perhaps not certain, may possibly be combined with the analysis of stylistic features to provide clues to the authorship of a piece of code.

In the following section, we have detailed some of the features that we believe to be the most useful in such a comparison. We believe that an in-depth study of these features in the code of many programmers may result in some useful forensic information.

2 Analysis of Unauthorized Code

We will consider two different cases where code remnants might be analyzed. These differ in the nature of the code that was left for analysis.

2.1 Analysis of Executable Code

Often, the remnant of an attack is a piece of executable code, such as a virus or worm program. Unfortunately, many of the features of the code that could have been used in analysis have been stripped away during compilation. Comments and indentation have been removed, and identifiers have been replaced by memory addresses or register names. Additionally, optimizations may have been performed on the code, possibly giving the executable code a very different structure than the original program source.

For example, an optimizing compiler might generate the same executable code for each of the following C language program fragments:

```
for (x = 0; x < 10; x++) {
        func(x);
}
```

```
x = 0;
while (x < 10) {
        func(x);
        ++x;
}
```

[1] `cstyle` was taken from the Usenet, author unknown.

```
x = 0;
while (TRUE) {
        func(x);
        if (x++ == 10) break;
}
```

```
x = 1;
do {
        func(x-1);
        x++;
} while (x <= 10);
```

The original source code might have actually been in Fortran:

```
      DO 15 X=0, 9, 1
         CALL FUNC(X+0)
15    CONTINUE
```

or in Pascal:

```
    for x := 0 to 9 do
        func(x);
```

Each of these different source code segments exhibits features that could possibly be used in identifying the style of programming of an individual. These features may be lost to the examiner of the resultant executable code.

For example, during the analysis of the Internet Worm program ([7, 19]), its remnant was reverse-engineered to C programs that compiled to identical binary versions. In many cases, the analysts chose arbitrary names for variables and local subroutines — the compiler would not save the values, so the choices did not matter. When the disassembled code was later matched against a copy of the "real" source code, many small differences with the reverse-engineered copies were observed that compiled to the same binary.

Executable code, even if optimized, still contains many features that may be considered in the analysis:

Data Structures and Algorithms Competence with, and preference for, certain data structures may be extracted from executable code. This may provide a clue to the background of the code author. For example, it is unreasonable to suspect a beginning programmer of authoring code that made extensive use of a B-tree for data storage. Similarly, the choice of algorithms used in a program may present features worthy of analysis. It seems likely to conclude that a programmer will continue to use algorithms with which they are particularly comfortable.[2]

As an example, consider the Internet Worm mentioned earlier. The code used linked lists as the primary data structure for building long lists that were repeatedly searched. This was certainly a poor approach, as the repeated searches of long lists dramatically reduced the efficiency of the program. This was noted in [19], and a correspondent later related that the Worm's author, Robert T. Morris, had been instructed in the Lisp programming language in his first undergraduate data structures and algorithms course.[3] Although a coincidence such as this is certainly not sufficient evidence upon which to base any specific action, it may help reinforce other conclusions, obtained through other means.

Related to this is the manner in which data structures are accessed. In languages with both pointers and arrays, the choice of which mechanism to use is often very programmer-specific. Likewise,

[2] Our experience with both undergraduate and graduate student programmers supports this supposition.

[3] Lisp is primarily a list-processing language.

using overlapped structures (the `EQUIVALENCE` statement in Fortran, and the `union` statement in C, for instance) provide a similar indicator. Some programmers use these structures, while others use coercion and bitwise operations to achieve the same goals.

Compiler and System Information Executable code may contain telltale signs of its origin. A unique ordering of the instructions may point to a specific compiler as the source of the code. The code may contain invocations of system calls found only in certain operating systems. These bits of information may rule out or support individuals as the author of the code.

In the case of many viruses, analysis of the binary code may reveal that it was written in C or Pascal from a certain vendor. This can be determined because support routines (sometimes known as "thunks"[1]) and library calls unique to that vendor are present in the binary.

Programming Skill and System Knowledge The level of expertise of the author of the program, with both the operating system in question and computer programming in general, may be estimated from the executable code. For example, programming that duplicates functionality already provided by standard system calls, makes extensive use of recursion, or makes proper calls to advanced system functions could indicate different levels of knowledge and skill.

Additionally, the inclusion or omission of error-checking code is also quite telling. Some programmers seldom (or never) include exception handling code in their programs. Others always include such code. In instances where the code is sometimes included, this may provide an identifiable set of routines that the author always checks (perhaps because of past program failures with those routines). This set could then be compared with the set from other, known programs as a metric of similarity.

Choice of System Calls The support functions used in the code may also indicate something about the background of the programmer. For instance, in the UNIX system, there are sometimes two different calls to locate the first instance of a particular character in a string. The `index` routine is derived from the Berkeley (BSD) version of UNIX, and the `strchr` function is derived from the System V version of UNIX. Users will usually exhibit a distinct preference for one call or the other when programming in an environment that provides both functions. Experience with reading and porting code has convinced us there are many such observable preferences.

Errors Programmers will usually make errors in all but the simplest or most carefully coded programs. Some programmers will consistently make the same types of errors, such as off-by-one errors in loops processing arrays.[4] Cataloging and comparing these faults may provide yet another metric for determining authorship of suspect code.

It is possible that the symbol table may still be present in the executable, as is often the case when the compiler is told to generate debugging information. In this case, several of the features normally associated with program source code may also be examined in the executable code.

2.2 Analysis of Source Files

Program source code provides a far richer base for writer-specific programming features.

Language Perhaps the most immediate feature of the code is the programming language chosen by the author. The reasons behind the choice may not be obvious, but could include availability and knowledge. It would be unreasonable to suspect an individual of being the author of a program written in a programming language that he does not know.

Formatting The formatting of source code often exhibits a very personal style. Format also tends to be consistent between programs, making it easier for an author to read what she has written. These factors indicate that the formatting style of code should yield writer-specific features. Placement of compound statement delimiters, multiple statements per line, format of type declarations, formatting

[4] This same tendency can be used in other contexts to direct software testing to likely faults.[4, 20]

of function arguments, and many other characteristics may be identified in the code in question. This assumes that the programming environment in question does not have a rigid, widely-used code formatter ("pretty-printer") that may have produced the observed style.

Another bit of information that could become available in this analysis is editor choice. For example, it may be possible to recognize the formatting styles produced by an editor such as Emacs, or to detect embedded mode-setting commands. Syntax-directed program editors may also provide a distinct and unusual style, should they become somewhat more common.

Special features Some compilers support *pragmas* or special macros that are not present on every system. The presence of any of these special features may provide clues as to the software development environment of the author. Inclusion of conditional compilation constructs, especially those involving initialization and declaration files, may also provide similar information about environment.

Comment Styles Users often tend to have a distinctive style of commenting their programs. Some use lines of a graphic character to set off comments from code. Others place comment headers above each function, describing it. Still others avoid comments at all costs.

The frequency and detail of the comments present may also be distinctive. Some programmers comment with short tags, and others write whole paragraphs. This may result in a measurable pattern.

Variable Names Choice of variable names is another aspect of programming that often indicates something about the author. Some programmers prefer to connect words in identifiers with an underscore, others take the SmallTalk approach and capitalize the first letter of each word with no separator. Ardent software engineers may use a naming scheme, such as Meta-Programming, that includes type information in the variable name.[17] Still others would never dream of using more than one or two characters in a variable name. A useful metric for identifier analysis might be something such as the distribution of Hamming distances between names.

Most experienced programmers have a set of "utility" variable names they use for local variables when coding small segments. Common examples include `junk`, `foo`, `temp`, `ii`, and `indx`. An analysis of these names may be useful in matching against other code by the same author.

Spelling and Grammar Many programmers have difficulty writing correct prose. Misspelled variable names (e.g., `TransactoinReciept`) and words inside comments may be quite telling if the misspelling is consistent. Likewise, small grammatical mistakes inside comments or print statements, such as misuse or overuse of em-dashes and semicolons might provide a small, additional point of similarity between two programs.

For example, a former colleague of one of us would consistently misspell forms of the word "separate." Thus, seeing a prompt in a program that read

```
Enter 3 values, seperated by a blank:
```

was a fairly certain indicator that he had written the code.

Use of Language Features The way in which authors make use of a programming language may also differentiate them. Some authors may consistently use a subset of the features available, while others may make more complete use of all features. For example, an author may consistently use a `while` loop, even when a `for/do` or `repeat..until` loop would be more appropriate. Similarly, the use of nested `if` statements in place of `case` statements, or the (lack of) specification of default options in `case` statements could be differentiating features of code.

Other examples that fall into this category include returning values in procedure parameters versus function return values, use of enumerated data types, use of subrange types, use of bitwise boolean operations, use of constant data types, and use of structures and pointers.

The average size of routines may also be used as an identifying feature: some programmers will code 300 line modules, and others will never have a module larger than will fit on the screen all at once. This, of course, is highly dependent on the overall size and complexity of the task being coded.

One aspect of use of language features relates to computer languages that a programmer may know best or learned first. For instance, programmers who spend most of their time using procedural languages seem to seldom use recursion. Learning programming in a language such as Basic or Fortran is also likely to lead to reduced use of `while` and `do ...until` structures. Further study of such influences may yield a discernable tendency to use or avoid particular language features.

Scoping The ratio of global to local identifiers may be an author-specific trait. Additionally, declaring helper functions as accessible only in a limited scope may also contribute to identification of the programmer.

Execution paths A common factor found when analyzing student programs and also when analyzing some malicious code (including [19]) is the presence of code that cannot be executed. The code is present either as a feature that was never fully enabled, or is present as code that was present for debugging and not removed. This is different from code that is present but not executed because of an error in a logic condition — it is code that is fully functional, but never referenced by any execution path.

As an example, consider the following section of code in the C language:

```
#define DEBUG 0
main() {

   /* some amount of code here */

if (DEBUG) {
    printf ... many debugging values here ...
}
```

In this example the code will never be executed. The manner in which it is elided leaves the code intact, and may provide some clue to the manner in which the program was developed. Furthermore, it may contain references to variables and code that was not included in working parts of the final program — possibly providing clues to the author and to other sources of code used in this program.

Bugs Some authors consistently make the same mistakes in their coding. Often, these are faults that only rarely cause problems, and then only with extremal values or when ported to other hardware. It is precisely because these bugs seldom cause problems that users tend to continue to introduce them into their code. The presence of identifying bugs should provide very strong evidence of similarity between two pieces of code.

As examples, we have noted the following in code by both students and colleagues:

- Failure to code bitwise operations to reflect different byte ordering on the target machine — the so-called "little-endian" vs. "big-endian" problem.
- Failure to check for numeric overflow or underflow, or assuming that the internal numeric representation was of a certain (different) form (cf. [20]).
- Assuming that uninitialized pointers can be dereferenced without generating a fault.
- Assuming the stack can hold very large value-copy parameter structures when doing subroutine calls.
- Failure to check error returns from some system calls that can (rarely) fail.

Metrics Software metrics might be employed to identify an individual's average traits. Some applicable metrics could include number of lines of code per function, comment-to-code ratio, function complexity measures, Halstead measures, and McCabe metrics.[5]

Clichés Programmers tend to write software in terms of well-understood algorithms and data structures known as clichés. Examples include sorted lists, binary searches, and hash tables. These clichés

implement higher level concepts, allowing the programmer to avoid reinventing the wheel with each new program. Software tools exist that will identify all occurrences of a set of clichés in a program.[16] Identification of the set of clichés commonly used by a programmer should prove useful as an identifying feature.

3 Application and Difficulties

It seems clear that there are many potential factors that could be examined to determine authorship of a piece of software. Ideally, this analysis would be used to identify a suspect, and then a search would be made of storage and archival media to locate incriminating sources. However, a more likely scenario would see a set of metrics and characteristics derived from the code remnant and then compared with representative samples written by the suspects. This comparison must be made with considerable care, however, to prevent complicating factors from producing either false positive or false negative indications.

Richard Bailey, in his studies of authorship forensics, puts forth three requirements to be met before authorship classification is attempted:

1. that the number of putative authors constitute a well-defined set;
2. that there be a sufficient quantity of attested and disputed samples to reflect the linguistic habits of each candidate;
3. that the compared texts be commensurable.[2, page 7]

Although these requirements were intended for classification of prose, they would appear to be directly applicable to the area of software forensics.

A likely complication, for instance, is the amount of code available for comparison. A small amount of suspect code (e.g., a computer virus) might not be sufficient to make a reasoned comparison unless very unusual indicators are present.

Another complication is the reuse of code. If the author has reused code from her earlier work, or code written by others, the effect may be to skew any metrics derived from the suspect code. It might be enough to correctly indicate *original* authorship, but that might not identify the actual culprit. In some cases, code reuse may be obvious and it may be omitted from the comparison. However, there may be cases where that is not possible. Likewise, if the suspect code was written as part of a collaboration, the characteristics of the individual authors may be subsumed or eliminated entirely.

A clever programmer, aware of this method, might disguise her code. This would probably involve using different algorithms and data structures than what she would normally use. Although this might eliminate the possibility of a match based on internal characteristics, it might also make the code more likely to fail in use. This should also make the programmer use more testing, and keep intermediate versions of the program that could later be matched against the suspect code.

There is also the potential that the underlying application may have a strong influence on the overall style and nature of the code. For instance, if we are attempting to match characteristics of a small MS-DOS boot record virus, and the code we compare against is for a UNIX-based screen editor, it is unlikely that we would find much correspondence between the two, even if they were written by the same author. Therefore, we must be certain that we compare similar bodies of code.

4 Concluding Remarks

There are many differences between handwritten prose and computer programs. Handwriting samples are usually fixed in an instant, and prose is usually not incrementally developed, while a program evolves over

time. Multiple changes to a section of code as a program is developed can lead to a structure that the author would have been unlikely to create under other circumstances.

Coding is also different in that code written by others is often incorporated into a program. Often, a program is not the result of the influence of only one author. We suspect that this would severely impair the selection of writer-specific code features without knowledge of the development of the program.

Nonetheless, if there is a sufficiently large sample of code and sufficient suspect code, if there are unusual features present, and if we have correctly chosen our points of comparison, this method may prove to be quite valuable. Currently, similar *ad hoc* methods are used by instructors when they compare student assignments for unauthorized collaboration (cheating). The samples are usually not big, but the characteristics are often distinctive enough to make valid conclusions about authorship. Developing and applying more formal methods should only improve the accuracy of such methods, and make them available for more in-depth investigations. Ad hoc methods have already been developed in analyzing some cases, such as the Internet Worm incident[19] and the WANK/OILZ Worms,[14] but improvements are certainly possible.

Not only would a formal method of *software forensics* aid in the determination of malicious code authorship, it would have other uses as well. For instance, determining authorship of code is often central to many lawsuits involving trade secret and patent claims. The characteristics we have outlined in this paper might be used to determine if code is, in fact, original with an author or derived from other code. However, a rigorous mathematical approach is needed if any of these kinds of results are to be applied in a court of law (cf. [18]).

We believe that if this approach is developed, it may also prove useful in applications of reverse-engineering for reuse and debugging. The analysis of code to determine characteristics is, at the heart, a form of reverse-engineering. Existing techniques, however, have focused more on how to recover specifications and programmer decisions rather than to determine programmer-specific characteristics (cf., [10]).

Further research into this technique, based on examination of large amounts of code, should provide further insight into the utility of what we have proposed. In particular, studies are needed to determine which characteristics of code are most significant, how they vary from programmer to programmer, and how best to measure similarities. Different programming languages and systems should be studied, to determine environment-specific factors that may influence comparisons. And most importantly, studies should be conducted to determine the accuracy of this method; false negatives can be tolerated, but a significant number of false positives would indicate that the method is not useful for any but the most obvious of cases.

Acknowledgments

Our thanks to Richard DeMillo for suggesting some related references of interest. Thanks to both Ronnie Martin and Tom Longstaff for their comments. We are grateful to Gene Schultz for his comments, and for encouraging us to commit our long-standing interest in this area to paper.

References

[1] Alfred V. Aho and Jeffrey D. Ullman. *Principles of Compiler Design*. Addison-Wesley, 1978.

[2] Richard W. Bailey. Authorship attribution in a forensic setting. In D. E. Alger, F. E. Knowles, and Joan Smith, editors, *Advances in computer-aided literary and linguistic research. Proceedings of the Fifth International Symposium on Computers in Literary and Linguistic Research*, pages 1–20, 1979.

[3] Louis J. Chmura and Henry F. Ledgard. *COBOL with Style: Programming Proverbs*. Hyden Book Company, Inc., Rochelle Park, NJ, 1976.

[4] B. J. Choi, R. A. DeMillo, E. W. Krauser, R. J. Martin, A. P. Mathur, A. J. Offutt, H. Pan, and E. H. Spafford. The Mothra tools set. In *Proceedings of the 22nd Hawaii International Conference on Systems and Software*, pages 275–284, Kona, HI, January 1989.

[5] S. D. Conte, H. E. Dunsmore, and V. Y. Shen. *Software Engineering Metrics and Models.* Benjamin/Cummings, 1986.

[6] The Cuckoo's Egg. *Clifford Stoll.* Doubleday, New York, NY, 1989.

[7] Mark W. Eichin and Jon A. Rochlis. With microscope and tweezers: an analysis of the Internet virus of November 1988. In *Proceedings of the Symposium on Research in Security and Privacy*, Oakland, CA, May 1989. IEEE-CS.

[8] L.W. Cannon et. al. *Recommended C Style and Coding Standards.* Pocket reference guide. Specialized Systems Consultants, 1991. Updated version of AT&T's Indian Hill coding guidelines.

[9] Joseph A. Fanciulli. The process of handwriting comparison. *FBI Law Enforcement Bulletin*, pages 5–8, October 1979.

[10] M. F. Interrante and Z. Basrawala. Reverse engineering annotated bibliography. Technical Report SERC-TR-12-F, Software Engineering Research Center, University of Florida, January 1988.

[11] Brian W. Kernighan and P. J. Plauger. *The Elements of Programming Style.* Mcgraw-Hill, second edition, 1978.

[12] V. Klement, R. Naske, and K. Steinke. The application of image processing and pattern recognition techniques to the forensic analysis of handwriting. In *1980 International Conference: Security Through Science and Engineering*, pages 5–11, 1980.

[13] Henry F. Ledgard, Paul A. Nagin, and John F. Hueras. *Pascal with Style.* Hayden, 1979.

[14] Thomas A. Longstaff and E. Eugene Schultz. Beyond preliminary analysis of the WANK and OILZ worms: A case study of malicious code. Technical Report UCRL-JC-108518, Lawrence Livermore National Laboratory, August 1991.

[15] Frederick Mosteller and David L. Wallace. *Applied Bayesian and Classical Inference: The Case of the Federalist Papers.* Springer Series in Statistics. Springer-Verlag, 1964.

[16] Charles Rich and Lina M. Wills. Recognizing a program's design: A graph-parsing approach. *IEEE Software*, 7(1):82–89, January 1990.

[17] Charles Simonyi. Meta-programming: A software production technique. *Byte*, pages 34–45, September 1991.

[18] Herbert Solomon. Confidence intervals in legal settings. In M. H. DeGroot and S. E. Fienberg J. B. Kadane, editors, *Statistics and the Law*, pages 455–473. John Wiley & Sons, 1986.

[19] Eugene H. Spafford. The Internet worm program: an analysis. *Computer Communication Review*, 19(1), January 1989. Also issued as Purdue CS technical report TR-CSD-823.

[20] Eugene H. Spafford. Extending mutation testing to find environmental bugs. *Software Practice and Experience*, 20(2):181–189, February 1990.

[21] Dennie Van Tassel. *Program Style, Design, Efficiency, Debugging, and Testing.* Prentice-Hall, Englewood Cliffs, NJ, second edition, 1978.

Some More Thoughts On The Buzzword "Security Policy"

David M. Chizmadia
Information Systems Security Organization, NSA
Ft. George G. Meade, MD 20755-6000
(410) 859-4463
Chizmadia@DOCKMASTER.NCSC.MIL

Keywords

Security Policy, Systems Development Process, Security Objectives, Security Policy Framework

Abstract

It is nearly axiomatic in the Information Security field that a "security policy" is the basis for defining and assessing security in an information system. Unfortunately, the precise meaning and characteristics of a security policy have never been defined by the community. This has led to a situation where individuals define "security policy" according to their background. This often makes communication about other basic information security issues more difficult than it has to be. Starting from the security policy frameworks proposed by Daniel Sterne and the ITSEC, this paper will propose a consolidated framework for security policy development and application. It will then postulate some ways to use this framework to better ensure that information technology reliably supports and reinforces the security objectives of an organization.

Introduction

The primary goal of this paper is to develop a practical and straightforward approach to using security policy to ensure that information technology reliably supports and reinforces an organization's security objectives. The approach taken is to synthesize a single set of definitions out of those proposed in Sterne and the *ITSEC*. Using these definitions, the paper will then discuss how policy can be used to shape the specification and guide the implementation of the information technology employed by an organization to protect its resources. The paper concludes with an analysis of some issues that need further study and exposition.

Background

One of the basic tenets in the information security community is that one cannot produce a convincing argument that a system is either trusted or secure without an explicit security policy from which to argue. All existing security evaluation criteria [1, 4, 6] require a "security policy" as part of the documentation that defines the entity being evaluated. However, as noted by Sterne [5], security policy is used in these criteria at least two different ways. The Trusted Computer System Evaluation Criteria (*TCSEC*) [6] glossary defines a security policy as "The set of laws, rules, and practices that regulate how an organization manages, protects, and distributes sensitive information." Sterne makes the point that in this sense, security policy constrains the actions of people. In apparent contradiction to its glossary definition, the *TCSEC* Security Policy Control Objective identifies a security policy as the "...statement of intent with regard to control over access to and dissemination of information..." that "...must accurately reflect the laws, regulations, and general policies from which it is derived." [6, p.

59] In this sense, security policy constrains the actions of computer processes. Both concepts are valuable to understanding security, but not when they are both identified by the same term. In [5], Sterne approached this problem by rigorously defining two types of security policy, Organizational and Automated, that were consistent with the *TCSEC* usages and which reinforced the *TCSEC* concentration on confidentiality. A similar set of rigorous definitions is also present in the Information Technology Security Evaluation Criteria (*ITSEC*) [4, paras 2.8-2.17].

Current Definitions for Security Policy

The TCSEC

The *TCSEC* provides two definitions for security policy. The first is the "formal" one found in the *TCSEC* glossary that defines the security policy as:

> "The set of laws, rules, and practices that regulate how an organization manages, protects, and distributes sensitive information." [6, pg. 115]

The second definition is found in the *TCSEC* chapter on Control Objectives. Here, the Security Policy Control Objective states:

> "A statement of intent with regard to control over access to and dissemination of information, to be known as the security policy, must be precisely defined and implemented for each system that is used to process sensitive information. The security policy must accurately reflect the laws, regulations, and general policies from which it is derived." [6, pg. 59]

The ITSEC

The *ITSEC* presents a fairly detailed framework for the notion of security policy. This framework recognizes three levels of security policy. The first level is that of the **Corporate Security Policy**, which consists of:

> "... (the) general security standards that apply to all systems within the organization and define the security relationship between the organization and the outside world. These standards can be considered to be the Corporate Security Policy: the set of laws, rules and practices that regulate how assets, including sensitive information, are managed, protected and distributed within the organization." [4, para 2.10]

The *ITSEC* authors then proceed to further explain the nature and utility of the corporate security policy, as well as start linking it to the next level of security policy by writing:

> "The primary responsibility of the Corporate Security Policy is to provide the context for the identification of system security objectives. Identifying relevant corporate assets, general threats, and the results from risk analysis will assist in the identification of these system security objectives." [4, para 2.11]

Security objectives are described in the *ITSEC* as the statements about desired security functionality that explain why the organization wants the functionality. That is, they express how the Target of Evaluation (*TOE*) is expected to contribute to the overall security of the system in which it is used. [4, para 2.2] The security objectives are a central component of the **system security policy**, the next level of security policy in the *ITSEC*. The System Security Policy is described in the following two paragraphs from the *ITSEC*.

"The System Security Policy specifies the set of laws, rules, and practices that regulate how sensitive information and other resources are managed, protected and distributed within a specific system. It shall identify the security objectives of the system and the threats to the system." ... "The System Security Policy shall cover all aspects of security relating to the system, including (the) associated physical, procedural and personnel security measures.[4, para 2.9]

"In the context of an individual system, the System Security Policy shall define the security measures to be used to satisfy the system security objectives in a way which is consistent with the Corporate Security Policy. The security measures required by the System Security Policy will be implemented by a combination of security enforcing functions implemented by the TOE, and by physical, personnel, and procedural means. The System Security Policy shall clearly indicate the division of responsibility between the security enforcing functions and the other means." [4, para 2.12]

The lowest level of security policy as defined in the *ITSEC* is the **Technical Security Policy**, which is described by the following paragraph:

"The IT (Information Technology) security measures of a System Security Policy may be separated from the remainder of the System Security Policy, and defined in a separate document: A **Technical Security Policy**. This is the set of laws, rules and practices regulating the processing of sensitive information and the use of resources by the hardware and software of an IT system." [4, para 2.13]

Although the Technical Security Policy is the most specific form type of security policy identified in the *ITSEC*, the *ITSEC* also identifies something called the **Product Rationale**, which is described in the following paragraph:

"The product rationale shall identify the intended method of use for the product, the intended environment for use of the product and the assumed threats within that environment. It shall include a summary of the products security features, and define all assumptions about the environment and the way in which the product will be used. This shall include personnel, physical, procedural and IT security measures required to support the product, and its dependencies on system hardware, software, and/or firmware not supplied as part of the product." [4, para 2.17]

Daniel Sterne

In [5], Daniel Sterne presents a framework for security policies that reinforces the notion of security policy presented in the *TCSEC* while at the same time making it more precise. The definitions of his framework are presented below to provide completeness in this background section.

"Security Policy Objective - A statement of intent to protect an identified resource from unauthorized use. The statement must identify the kinds of uses that are regulated. The identified resource must be tangible or have some form that is tangible. A security policy objective is meaningful to an organization only if the organization owns or controls the resource to be protected.

"Organizational Security Policy (OSP) - The set of laws, rules, and practices that regulate how an organization manages, protects, and distributes resources to achieve specified security policy objectives. These laws, rules, and practices must identify criteria for according individuals authority, and may specify conditions under which individuals are permitted to exercise or delegate their authority. To be meaningful, these laws,

rules, and practices must provide individuals reasonable ability to determine whether their actions violate or comply with the policy.

"Automated Security Policy (ASP) - The set of restrictions and properties that specify how a computing system prevents information and computing resources from being used to violate an organizational security policy. To be considered an ASP, there must exist a persuasive set of engineering arguments that these restrictions and properties play a key role in the enforcement of an organizational security policy." [5]

The Security Policy Hierarchy

The above definitions, and the rationales used to support them, were analyzed and compared to develop a framework for using security policy as a design and development tool. During the comparison, three distinct levels of security policy abstraction emerged. These levels and their relationship to the *TCSEC*, the *ITSEC*, and Sterne are shown in Figure 1.

TCSEC	ITSEC	Sterne	Proposed Levels
Security Policy (Glossary)	Corporate Security Policy Security Policy Objective	Security Policy Objective Organizational Security Policy	Corporate Vision of Security
Security Policy (Control Objective)	System Security Policy Technical Security Policy	Automated Security Policy	Corporate Philosophy for Secure Use of Information Technology
	Product Rationale		Information Technology Security Services

Figure 1: Comparison of Security Policy Hierarchies

The first level is that of the corporate vision for security. At this level the concern is largely with tangible resources and takes the form of: restrictions on the individual actions and group interactions of people with respect to the use of corporate resources, the procedures for granting or revoking authorization to use resources, and the rules for tracking and reviewing compliance with the constraints and authority. The next level is that of the corporate philosophy for securely using information technology: that is, proactively using information technology support the vision for security. At this level, the corporate vision is interpreted to define both the role of information technology in enforcing the corporate vision and the responsibilities of the information technology users to properly use or maintain the technology. Finally, there is a level where the IT is viewed as a self-contained universe within which specific services are provided for controlling the interactions of the entities inhabiting that uni-

verse. The security policy at this level is unique in that it is tied to the IT rather than the organization and is therefore independent of the specific policy of an arbitrary organization.

Starting with these three general levels of abstraction the following framework of definitions was developed to provide a unified set of referents for discussing security policies.

Corporate Vision of Security

Security Policy Objective

> A statement of intent to protect an identified resource from unauthorized use. The statement must identify the kinds of uses that are regulated. The resource must be tangible, or be a derivation of something that is tangible The statement should also identify why the resource is valuable and the effect of its unauthorized use on the organization. A security policy objective is meaningful to an organization only if the organization owns or controls the resource to be protected.

This definition is drawn largely from the definition given by Sterne [5], except for two changes to make it more general and useful. The first change was to modify the restriction that resources have a tangible form. The intent of this requirement, i.e., linking the resource to the "real world" is desirable, but the specific wording was overly restrictive. For example, in the case of a credit bureau, the main resources to be protected are the credit histories it maintains; however, each history is only information - one can't really trace it back to the people or real money (neither are under the control of the credit bureau). The second change supports the ITSEC notions of risk analysis and threat identification by providing a clear starting point for assigning risk and identifying threats. This framework does incorporate Sterne's view that the organizational policy is derived from the *security policy objectives* , rather than the *ITSEC* view that the reverse is true.

Organizational Security Policy (*OSP*)

> The set of laws, rules, and practices that regulate how an organization manages, protects, and distributes resources, to achieve specified *security policy objectives*. These laws, rules, and practices must identify criteria for according individuals authority, and may specify conditions under which individuals are permitted to exercise or delegate their authority. To be meaningful, these laws, rules, and practices must provide individuals reasonable ability to determine whether their actions violate or comply with the policy.

The words of this definition are the same as those for Sterne's Organizational Security Policy, which capture the essence of the *ITSEC* definition for a corporate security policy, with the caveat noted above about the different perspectives on the security objectives. This framework retains the perspective of both Sterne and the *ITSEC* that the *Organizational Security Policy* is concerned with the relationships between people, the operating practices of the organization, and the "real" resources that are important to the organization.

Corporate Philosophy for Securely Using Information Technology

Once an organization has identified its security policy, it may then find itself in a situation where some aspects of the management, protection, and distribution of its resources are actually done with the assistance of information technology. At this point a top-down approach to security suggests that the organization should develop another level of policy that sets out corporate interpretation of the *organizational security policy* with respect to how information technology will be used across the organization such that security is not compromised. This level of policy is defined now as the:

Organizational Security Policy for Information Technology (*OSPIT*)

> Interpretations of the (sub)set of laws, rules, and practices in the *OSP* that are implemented or enforced with the aid of information technology. These interpretations must address: how specific aspects of resources defined in the security policy objectives are represented electronically, the

role of the information technology used to protect those electronic representations, the additional non-technical procedures (e.g., physical, procedural and personnel security measures) that lead to the appropriate use of the information technology to accomplish the identified security objectives, and the operation and maintenance of the information technology.

The *OSPIT* is closely akin to the *ITSEC* Corporate Security Policy, which has a definite focus on the information technology aspects of security. It is important to note that the *OSPIT* is developed with a clear view towards providing cost-effective protection for corporate resources. As such the final *OSPIT* will reflect a practical approach to trading off procedural approaches to security versus automated ones based on relative cost and effectiveness.

In a moderately large to large organization, there may be several distinct configurations of information technology, called systems in the *ITSEC* and **information technology systems** in the remainder of this paper. These information technology systems could be used to manage, protect, or distribute different types of information in support of different subsets of the security policy objectives. For example, a manufacturing company may have separate systems for assembly line control, accounting, and office automation. To ensure that each information technology system supports the appropriate *organizational security objectives*, there must be a clear statement that relates the specific information resources that it controls back to specific objectives and policy directives. This statement might take the form of an interpretation of the *OSPIT* for the information technology system. Such a statement is defined now as the:

Information Technology System Security Policy (*ITSSP*)

The set of restrictions and properties that both specify how a specific information technology system prevents its storage, communications, and computing resources from being used to violate an *organizational security policy* and describe the complementary procedures that people using the ITS must follow to ensure that the information the information technology system uses to make decisions corresponds to reality. The *ITSSP* must be accompanied by a persuasive set of engineering arguments that its restrictions and properties play a key role in the enforcement of an organizational security policy.

Information Technology Security Services

The *ITSSP* corresponds to Sterne's Automated Security Policy (ASP), in that it is the link between a specific ITS and the organizational security objectives that it supports. Sterne identifies the ASP as the lowest level of security policy, on the premise that policy in the absence of purpose is not very useful. This paper takes a slightly different view.

For compelling economic reasons, most information technology systems today are comprised primarily of commercially available products. The vendors of these products identify general market segments that have similar requirements and develop a product that directly satisfies most of those requirements and can be tailored to satisfy the rest. The performance characteristics of these products are expressed in general rather than specific terms. For instance, a CPU's speed is measured in instructions per second rather than time to recalculate the Bank of America's quarterly earnings spreadsheet. The same is true for the security features of products. The security characteristics of the product are expressed in terms of the abstract resources it controls (e.g., files, network connections, database views, spreadsheet cells, document paragraphs, etc); the types of control it provides (e.g., read, execute, write, delete, etc); and the assumptions about the environment in which the product exists that must hold true for the controls to be effective. This expression of a product's security characteristics is defined here as the:

Information Technology Protection Policy (*ITPP*)

The set of restrictions and properties that apply to the allocation and use of the storage, comput-

ing, and communication resources defined at the external (e.g., user, network physical port, system call, etc) interface to a specific configuration of information technology.

The term *Protection Policy* was chosen deliberately in order to distinguish between the *protection* features provided by the product for identified electronic resources and the actual *security* that those features *may* provide in a specific environment (i.e., with respect to a specific *ITSSP*). The *ITPP* corresponds closely to the Product Rationale of the *ITSEC* [4] or the Technical Security Policy of the *TDI* [7].

Using Security Policy In The System Development Process

The fundamental objective of the framework developed here is to encourage the appropriate use of abstraction in the system development process. A potential trap in system development is to "jump ahead" of oneself and go into too much detail for the effort at hand. Systems acquisitions are often driven by a desire for the technology most recently announced in major computer magazines, rather than a careful identification of current and future business activities followed by a detailed, and repeatable, analysis of which activities can be effectively automated.

A short-circuited systems development process often results in the use of one of two opposite, and equally undesirable, approaches to information system security. The first, and most common, approach is to treat security as an afterthought and go buy the fastest-smallest-newest technology available: disregarding the presence of, e.g., time-sensitive, national security information on the system with users at three levels of clearance. The second approach is to sieze the moral high ground and mandate high-grade security, even if the system is essentially just the parts catalog for army uniforms. In both cases, the purchase decision is being made on the basis of desire rather than need.

Although security policies are not the final solution to this problem, they do provide both management and technical personnel with a tool for identifying and tracking the fundamental security properties needed in the system. They also provide a tool to properly assign both responsibility and accountability for security in the organization as a whole and in the organization's information technology systems specifically.

Figure 2 shows how the framework proposed here relates to the general approach to system development advocated by most computer professionals. Any system development starts with a mission, or more specifically a perceived need. In the realm of security, this need corresponds to the *security policy objectives*, which are themselves an expression of the security needed for the mission. The mission sponsors then proceed to identify the mission requirements in terms of everything required to perform the mission and the approach to accomplishing the mission. This corresponds to the *organizational security policy*, which lays out all of the responsibilities for security in accomplishing the mission. Next, the mission sponsors identify a role for information technology in the mission and develop a plan for its use in the context of the mission. This phase of system development corresponds to the both the *Organizational Security Policy for information Technology* and the *Information Technology System Security Policy*. The *OSPIT* and the *ITSSP* may be one in the same if only one system is required to accomplish the mission. If there are multiple systems employed to accomplish the mission, however, there will be gradations of management that must be reflected by the security policies. Finally, actual technology is procured to accomplish the automated aspects of the mission. This technology may be general-purpose, commercially available technology or technology that is designed and built specifically for the mission. In either case, there should be a corresponding *Information Technology Protection Policy* that describes what shared resources the technology manages and how it provides the means to protect those resources from being shared unintentionally. The final step in the system develop process is to integrate the technology into the mission. In the context of the framework, there is no need to have a separate policy for this level because the primary purpose of all of the policy levels identified is to make sure that there are no surprises when this step is completed.

STAGES OF SYSTEM DEVELOPMENT	LEVELS OF POLICY
Identify Mission	Security Policy Objectives
Identify Requirements	Organizational Security Policy
Allocate Requirements to manual and automatic procedures and processes	Organizational Security Policy for Information Technology Information Technology System Security Policy
Procure Technology	Information Technology Protection Policy
Integrate Technology into mission activities	

Figure 2: Relationship of the Policy Framework to the System Development Process

This framework can also be understood in the context of organizational management, as well as system development. From this perspective, the expression of the *Security Policy Objectives* and the *Organizational Security Policy* is the responsibility of the highest tiers of management in the organization, since it establishes a corporate statement of the resources that are important and the individual responsibilities that each member of the organization has for protecting those resources. From a management standpoint, the effort applied is a useful management tool in and of itself, since it causes the managers who do it to consider carefully the basic goals and mission of the organization.

Once developed, the *OSP* becomes the requirements document for the managers in the organization who are charged with identifying the aspects of the organizational operations that can be cost-effectively augmented with information technology. As part of their planning and analysis activities, they would either produce an *Organizational Security Policy for Information Technology* (if there are different types of technology used to address different business objectives) or an *Information Technology System Security Policy* (if all of the organization operates within one ITS). When there is an *OSPIT*, it becomes the requirements document for the managers in charge of information technology for each of the unique business units, which would in turn develop the *ITSSP* for their ITS.

Properly formulated, the *ITSSP* is the specification for the security requirements needed in a specific ITS. The purpose of a good specification is to tell the developer or maintainer of an ITS what is needed, without dictating how to meet the need. Since the *ITSSP* consists of a set of restrictions, which provide a basis for determining the boundary of the ITS, and a set of properties, which identify the testable requirements that must be met at that boundary, it should (in theory) provide the perfect format

for specifying security requirements. The major impediment to this use is the absence of a well-known and precise format for presenting and interpreting an *ITSSP*. As will be observed again in the summary, this is a problem shared by all of the types of security policy and is one of the most important topics needing further study.

Postulating for the moment that one has developed a suitably concise and complete *ITSSP*, the final piece of the system development puzzle is to actually build an information technology system that satisfies the *ITSSP*. In the current environment, this is largely a matter of art: with the final product depending to a great extent on the experience and dedication of the individuals developing the system. The job involves taking the *ITSSP*, ensuring that the hardware and software that is developed or assembled in response to other mission requirements enforces the required security constraints, and producing a convincing argument that it all works together effectively.

Part of this development effort involves the derivation of an architecture for the technology that will satisfy the requirements. The components of this architecture will each satisfy a subset of the requirements for the ITS. Some of these components will have a responsibility for the overall ITS security. Often, there exist commercial products that provide some or all of the services needed. When this is the case, it would be extremely helpful to the developers if there were some way they could compare the coverage and the strength of the security services provided by the various "commercial off-the-shelf" products that are under consideration. The *ITPP* is a tool to satisfy this need by providing a statement of the product's security features, phrased in a fashion that corresponds to the requirements that the overall ITS must satisfy.

One of the goals of the *TCSEC* [6] was to provide exactly this service. Unfortunately, many feel that this benefit has not been realized. One reason for this is perhaps the mismatch between the policies evaluated with respect to the *TCSEC* and those required by specific systems. In the terms defined here, it is clear that the *TCSEC* evaluations done under the auspices of the NCSC are with respect to an *ITPP*, since they are tied more closely to the technology than to an operational mission. However, the words in the TCSEC imply that the policies evaluated will be *ITSSPs*.

An additional complaint about the *TCSEC* is that even when one understands the type of policy being evaluated, the range of such policies allowed by the *TCSEC* is too limited. More recent criteria [1, 4] have sought to address this problem. The *ITSEC* [4] approach to the problem is to allow a product vendor to define the product policy and show that the features provided enforce that policy. The *CTCPEC* [1] approaches the problem by defining specific requirements for the various security services possible in current technology and then letting a product vendor define the policy that justifies those services.

It can be observed that all three criteria share in common the characteristic that they do not provide a standardized format for the *ITPP* associated with evaluated products, nor do they require that these policies be considered part of the user documentation. This leads to the conclusion that any criteria lacking of these two requirements will not satisfy its primary customers, because they either will not get the *ITPP* at all or will not get it in a form that is easily compared with similar products.

Conclusion

Topics For Further Study

As was noted above, rigorously defining a framework for different levels of security policies only solves part of the problem. Unless each level of policy are expressed in a way that facilitates an understanding of the underlying requirements, nothing is gained. Developing guidance for the derivation and expression of each level of policy is the most crucial future effort in this discipline. A subsidiary effort that could be done in a shorter timeframe would be to identify some of the general classes of resources and their associated types of use that are commonly found in each type of policy. This would be a natu-

ral side-effect of the general policy expression research and would allow organizations to start expressing at least their security policy objectives in a consistent fashion.

Another major topic, primarily of interest to the Federal Government initially, is to relate each type of security policy to the formal systems acquisition process that has been defined for Federal procurements. As the guidance for expressing policies becomes more concrete, it could then be institutionalized into this process.

Summary

This paper has reviewed the contents of two major information technology security evaluation criteria - the *TCSEC* and the *ITSEC* - and the work of Daniel Sterne regarding security policies and has synthesized a security policy framework that incorporates all of the ideas found in the other works. This framework was then related to the various stages of the generic system development process. Finally, a set of topics that require further study and exposition were identified.

References

[1] *Canadian Trusted Computer Product Evaluation Criteria (Version 2.1e)*, Canadian System Security Centre Communications Security Establishment Government of Canada, December 1990.

[2] *Computer Security Policies: Challenges And Prospects*, Eugene V. Epperly, Proceedings of the Fifth Seminar on the DoD Computer Security Initiative (pp. 99-137), May 1982.

[3] *Evaluation, The DoD Certification Process, And Its Relation To The Trusted Computer System Evaluation Criteria*, William Neugent, Proceedings of the Sixth Seminar on the DoD Computer Security Initiative (pp. 83-86), November 1983.

[4] *Information Technology Security Evaluation Criteria—Version 1.2*, Senior Officials Group—Information Systems Security (SOG-IS), an EC advisory group, June 1991.

[5] *On The Buzzword "Security Policy,"* Daniel F. Sterne, Proceedings of the 1991 IEEE Computer Society Symposium on Research In Security And Privacy (pp. 219-230), May 1991.

[6] *Trusted Computer System Evaluation Criteria*, United States Department of Defense 5200.28–STD, December 1985.

[7] *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria*, National Computer Security Center, NCSC-TG-021 Version-1, April 1991.

STANDARD CERTIFICATION - PROGRESSION

Captain Charles R. Pierce

Air Force Cryptologic Support Center (AFCSC/SRP)
San Antonio, Texas 78243-5000

ABSTRACT

This paper presents the results of the first year's efforts of the Certification Working Group (CWG) formed under the Joint Logistics Commanders' (JLC) Information Systems Security (INFOSEC) Management Panel (IMP). The IMP CWG is developing a standard process for certifying computer systems. It is also developing education and training requirements for program managers and system developers. In the last year the CWG has developed a process for integrating risk management, certification, trusted product evaluations, and accreditation into the system life cycle. It has proposed training standards for systems' developers and education requirements for program management personnel. These standards are being translated into courses and curricula by a corollary Security Education and Training Working Group (SETWG).

STANDARDIZED CERTIFICATION

A paper, Standardized Certification [1], presented at the 14th National Computer Security Conference proposed the development of a standard for DoD Information Systems Security (INFOSEC) certifications. It presented some problems existing with current certification methodologies and some recommendations to remedy those problems. Among those recommendations was a certification standard. The paper recommended the standard:

1. Formulate a strategy for developing the standard certification process. The strategy would include plans for developing the process, acquiring the resources to implement it, education and training for personnel with certification responsibilities, proficiency standards for those personnel, and guidance for implementing the process in an operational environment.

2. Encompass all types of systems, including various small, embedded, and other "unique" types of systems.

3. Highlight when users or developers must make decisions to strictly apply the process or that a point has been reached where the process could not be adequately applied and a risk assumption decision must be made.

4. Clearly define who each process step applies to (user, developer, both, etc.). The process should contain some specifics not currently addressed, such as the life cycle maintenance of trusted software.

5. Describe where within the process various guidance such as NCSC Criteria or Technical Guidelines is to be applied and how.

6. Describe how, why, when, and where to use and accept other-agency evaluations, such as NCSC evaluations, or certifications from other DoD components.

7. Define when products from outside the certifying agency are specifically required, such as NCSC evaluated TCBs. Also included would be advice as to what documentation to use or require from those other evaluations, i.e., commercial documents or system unique documents.

8. Describe the applicability of DoD documents, service regulations and technical guidelines, particularly for contracted developments.

9. Must use currently available resources for implementing the process.

10. Must be mandated as a standard for all systems by inclusion in agency policy and regulations. Include the process in procurement policy, security certification, accreditation, and technical guidelines. Each program, off-the-shelf standard system, etc., must include a system-tailored process description in their security plan.

The paper also recommended that other guidance be completed to complement the certification standard. This would include how to use other agencies' evaluations as supporting certification documentation and defining responsibilities for recurring life cycle reviews and recertifications. Evaluation criteria should be developed or expanded beyond the DoD 5200.28-STD, e.g., for embedded systems, complex systems, real time systems, the Trusted Database Interpretation (TDI) [2], the Trusted Network Interpretation (TNI) [3], and applications software. A subsequent activity is to complete translation of these criteria into acquisition specifications formats and operational implementation guidance.

The paper encouraged the services and comparable agencies to improve their capabilities to provide certification support to systems (existing and developmental) by developing a program to educate people in multiple INFOSEC disciples, i.e., COMSEC, TEMPEST, etc.

Another Joint Logistics Commanders sponsored group, Computer Resources Management (CRM), hosted the San Antonio I workshop with one of its security goals to identify methods to facilitate certification and accreditation of multilevel systems and tools for certifying software systems. The products requested by the CRM were to facilitate accreditation and provide guidance for contracting security requirements. [4] The Security Panel IV

report from the San Antonio I workshop recommended the development of certification standards as an integral part of mission critical computer resources and automated data processing systems development. The panel also recommended that policies be established for implementing the standards and education and that training be required for those responsible for security throughout a system's life cycle. [5]

THE CERTIFICATION WORKING GROUP

Concurrent with the Standardized Certification paper's development and the San Antonio I workshop, the IMP CWG was formed. The IMP CWG is chartered to "...maximize the coordination and sharing of certification techniques, information, equipment, automated tools, and other resources in service to determine the best criteria, methodologies, and procedures to standardize the certification process across the military services and DoD agencies."

The IMP CWG was to initially produce a strategy for developing a certification standard process. The process was to be flexible enough to address the full range of system types and information sensitivity from unclassified to the highest classified and multi-compartmented information. It would address every life cycle stage of the system or network to be certified and the complexity of the system or network architecture to be certified. At a minimum, stand-alone or multi-user architectures, local area networks (LANs) and wide area networks (WANs) were to be addressed. [6]

The Standardized Certification paper was not intended to be a foundation for the IMP CWG, but it's apparent they have similar goals. The IMP CWG first met in April of 1991 and developed the strategy for meeting its charter requirements. Figure 1 represents the strategy plan and actions status to date. The IMP CWG has grown to include widespread membership (Figure 2) and meets at least quarterly, but lately more often since it is entering the most productive stages of the strategy plan.

THE STANDARD CERTIFICATION PROCESS

The first product developed by the IMP CWG was an identification of the key concepts and objectives [7] to guide the standard process development. This document also provided a subset of the standard terminology and definitions the IMP CWG would use during the process development. The first major point of agreement was to endorse the initial use of the NSTISS Instruction No. 4009 [8] as a standard INFOSEC glossary. NSTISSI No. 4009 defines certification as a:

> "Comprehensive evaluation of the technical and nontechnical security features on an AIS and other safeguards, made in support of the accreditation process, to establish the extent to which a particular design and implementation meets a set of specified security requirements."

Action	Begin	End
1. Identify key concepts and objectives, provide definitions	31 Jul 91	11 Feb 92
2. Identify IMP CWG needed resources	13 Sep 91	19 Nov 91
3. Define current risk management process	19 Nov 91	20 Aug 92
4. Define current evaluation process	19 Nov 91	29 Apr 92
5. Define current accreditation process	19 Nov 91	20 Aug 92
6. Develop integrated process	4 May 92	Apr 93
7. Define related processes revision requirements	4 May 92	Apr 93
8. Define recommended certification standards and policy	11 Feb 92	Apr 93
9. Develop certification standard	11 Jun 92	Apr 93
10. Develop policy recommendations	11 Jun 92	Apr 93
11. Develop process tailoring	11 Jun 92	Apr 93
12. Develop marketing program	5 Feb 92	20 Aug 92
13. Identify training requirements	11 Feb 92	20 Aug 92
14. Identify implementation resources	11 Jun 92	20 Aug 92

Figure 1. IMP CWG Development Schedule

The Standardized Certification paper recommended the development of a central point of sharing certification information and experience, including worked systems and lessons learned. Since this function was part of the NCSC's original charter, it agreed that this would be a logical function for it to eventually maintain. The IMP CWG decided to perform a similar collation of this information and temporarily maintain it until the standard was complete, then transfer the library to NCSC for life cycle maintenance. Reference to and use of the central library would become part of the certification process.

To define the current state of certification in DoD, the IMP CWG began by defining processes relative to certification, i.e., risk management, technical evaluation, and accreditation, taking the "best" from existing methodologies. These processes will be

Air Force Cryptologic Support Center, San Antonio TX
Sacramento Air Logistics Center, McClellan AFB CA
Air Force Materiel Command, Wright-Patterson AFB OH
Defense Investigative Service, Washington DC
US Marine Corp, Quantico VA
Electronic Systems Center, Hanscom AFB MA
Joints Chiefs of Staff, Washington DC
US Navy/Naval Electronic Security Center Washington DC
National Computer Security Center, Ft George G. Meade MD
National Security Agency, Ft George G. Meade MD
Naval Warfare Systems Command, Washington DC
Naval Electronic Systems Security Center, Washington DC
US Army/HQ CECOM, Ft Monmouth NJ
Mitre Corporation, Bedford MA
National Institute of Standards and Technology, Gaithersburg MD

Figure 2. IMP CWG Membership

refined into consistent flows and then integrated into the DoD development life cycle as defined in DoD Instruction 5000.2 [9]. Each process description includes the activities involved in that process, their development or product delivery timing in the life cycle, and the responsible agent for its production. After the "best of current technology" was defined, it is being integrated into the standard certification process.

In addition to the best of current technology, the process is including the best of current guidance development efforts. At appropriate points the user will be directed to use this type of guidance. Among these products are the NCSC-TG-24, An Introduction to Procurement Initiators on DoD Computer Security Requirements, Volume I-IV (Draft) [10], for developing specifications for trusted products acquisitions or AFSSM 5024, Security in Acquisitions (Draft) [11], for developing a Request for Proposal (RFP) for secure systems. Among the mandatory items a program will be required to develop are definitive system security policies, DAA-validate security requirements, certification and accreditation plans, risk analysis plans, and security test and evaluation plans. Ideally, if a Program Management Office (PMO) tailors the standard to meet its specific program roles, responsibilities, and schedule the result will be a consistent master certification and accreditation plan.

Although they are not part of the standard itself, other products will accompany it. Training requirements for relevant positions are being provided to the SETWG who is developing education curricula and training standards matched to the process. The first source for the resultant education and training should be the DoD schools such as the Defense Systems Management College, the Air Force Institute of Technology, and the various training schools. Academia is encouraged to use the curricula for developing similar training and undergraduate and graduate programs.

Also accompanying the standard will be tailoring guidance to match the process to system types other than the standard AIS. This includes such systems as embedded weapon systems, completely commercial-off-the-shelf (COTS) products, and integrated trusted and untrusted components. The tailoring guidance will provide for out of life cycle entry into the certification process, such as for a completely COTS purchase. This type of program could pass from requirements definition and enter the life cycle in the Production and Deployment [9] life cycle phase. The affected PMO, or equivalent organization, must know how to "catch up" such activities as security policy development or security testing that normally would have been done in earlier life cycle phases. The tailoring guidance will also address embedded and complex system issues. These include integrating products with varying certification schedules or trust levels into a system level certification process.

The standard will not provide guidance for matching various development models, such as the spiral model, to the process. This will be left to others, as will matching various certification implementation methodologies. The standard will be flexible enough so that most applied methodologies will satisfy it. All types of risk analysis tools, acquisition strategies, certification methodologies, or maintenance plans should fit.

The first edition of the standard process has been produced for internal review by the IMP CWG. It outlines the procedures and products that will be developed in each development phase. It provides mandatory points (primarily the standard development Milestones) by which these products must be available for DAA review and approval. To encourage the process' use, the standard uses the presentation style of DOD Instruction 5000.2. Delivered with the final process will be policy recommendations, including what changes should be made to existing service or agency policies to facilitate the process' implementation.

The process standard incorporates recommendations from many sources in addition to those mentioned above. Several features of the process are derived from the above recommendations. Among these are fully defined roles and responsibilities. These are allocated to position tiles, such as Program Manager or the Designated Approving Authority (DAA). It's impractical to be too specific because it's desired to make the process highly adaptable. For instance, in some cases, such as a small off-the-shelf purchase, all positions and responsibilities may be allocated to a single individual.

One of the most important development activities is early DAA designation. In many current developments the DAA was sometimes not designated until well into the development life cycle. Many critical decisions were made by someone other than the real DAA, leading to subsequent misunderstandings or problems. The standard clearly defines DAA appointments and responsibilities.

Early in 1992, the JLC directed the IMP to develop a marketing program for IMP developed products, including those from the SETWG as well as the IMP CWG. The IMP CWG was passed this direction and developed a marketing strategy that included conference presentations and articles for various publications with INFOSEC interests. Additionally, IMP CWG members will undertake efforts to keep their own agencies informed on the IMP CWG's progress and should be contacted for the current status of the standard's development.

The Director of Information System Security in the Office of the Deputy Assistant Secretary of Defense (Counterintelligence & Security Countermeasures) expressed a desire that standard certification and accreditation policy be issued the fall of 1992. The ODASD (CI&SCM) leads the Defense-wide Information Systems Security Program (DISSP), a joint effort of the National Security Agency, the Defense Intelligence Agency, and the Defense Information Systems Agency to develop standards for INFOSEC. The DISSP recommended that the IMP CWG merge its efforts with those of a similar group formed at NSA and form a DoD level working group. This merger began in May of 1992. The IMP CWG recommended to the JLC that policy development responsibility be passed to the DoD group as soon as possible. Development of the standard process will continue under the IMP CWG because it had responsibility to meet its JLC milestones and was well along its production schedule. IMP products will be harmonized with and passed to the DoD working group for enclosure in its processes.

THE FUTURE

The tailoring guidance for the process is being adapted for all types of systems and will be flexible enough to be modified for any type of development or acquisition. This includes everything from the smallest real time embedded processors to the largest integrated networks. It will fit within any acquisition strategy, e.g., either totally off-the-shelf systems or with lengthy development cycles.

The range of systems and information sensitivity being addressed involves that from totally unclassified to sensitive information with multiple compartments. The process will be simple enough to address common office automation. It will be complex enough to provide for integrating intelligence processors into collateral information systems in multilevel modes. The intent is to provide some guidance on the level of effort to be expended in the certification effort.

This proposed flexibility and adaptability is provided by developing a comprehensive model and then providing the instructions for tailoring it to a wide variety of environments. The IMP CWG is seeking prototypes, working systems, developments, or worked examples for applying the process during its evolution. The standard will not be proposed as policy without sufficient

guidance for tailoring and implementing it. The guidance provides a program manager with the ability take the full process and adapt it to his program's particular schedule, staffing, and other requirements or constraints.

One of the most difficult aspects of the standard will be to define criteria for across-agency acceptance of certifications. A simplistic view states that if a system was developed and certified according to the standard, the certification should be acceptable to other DoD agencies. The accepting agency must still review modification may to the standard process and the methodology used to implement the process. The data base of lessons learned, etc., could with time become a de facto set of "approved methodologies," based on experience.

Experiences gained coupled with the education and training products should lead to one of the IMP's longest term goals, producing a cadre of certified certifiers. These certifiers would review the efforts of system developers and provide standard certification assurances without performing the actual certifications themselves. Standards for certification performance will be coupled with the technical standards to then produce truly standard certifications.

A draft of the proposed standard should be available for community review by late 1992. The draft will be a minimum process with the intent to stay that way. There will be temptation to expand the standard to include points from various favored methodologies. This will be avoided so that each of these methodologies can fit in the standard. The immediate follow-on effort will be to develop language for implementing the standard in contracts.

CONCLUSION

The IMP CWG's goal is to have substantially completed its work on the certification standard by the middle of 1993. The ultimate product is to be submitted as either a DoD or Military Standard. The accompanying guidance could be either a DoD level instruction or manual. Recommendations on collateral policy changes will be made by working group members to their respective DoD components. We look to industry and the academic and training communities to develop programs for spread knowledge about the standard. The vendor community will be encouraged to develop models can that be adapted to the standard as well as various methodologies for implementing it.

REFERENCES

1. Pierce, C.R., Standardized Certification, Proceedings of the 14th National Computer Security Conference, 1991.

2. NCSC-TG-21, Trusted Database Management System Interpretation of Trusted Computer System Evaluation Criteria, 22 August 1990.

3. NCSC-TG-05, Trusted Network Interpretation of Trusted Computer System Evaluation Criteria, 31 July 1987.

4. San Antonio I Handbook, DoD Software for the 1990s, 30 November 1990.

5. San Antonio I Software Workshop Proceedings, "DoD Software for the 1990s," 11 December 1991.

6. Charter for Joint Commanders Group Communications-Electronics COMPUSEC Implementation Management Panel (CIMP) Certification Working Group (CWG), 6 March 1991.

7. Smyth, Mike, Certification and Accreditation (C&A): Goals and Objectives, Key Concepts, and Terminology, Computer Security (COMPUSEC) Implementation Management Panel Certification Working Group, 24 September 1991.

8. NSTISS Instruction No. 4009, National Information Systems Security (INFOSEC) Glossary, National Security Telecommunications and Information Systems Security Committee (NSTISSIC), 5 June 1992.

9. DoD Instruction 5000.2, Defense Acquisition Management Policy and Procedures, 23 February 1991.

10. NCSC-TG-24, An Introduction to Procurement Initiators on DoD Computer Security Requirements (Draft), 25 October 1991.

11. AFSSM 5024, Security in Acquisitions (Draft), 1 April 1992.

A TAMPER-RESISTANT SEAL FOR TRUSTED DISTRIBUTION AND LIFE-CYCLE INTEGRITY ASSURANCE

Mark Bianco
Hughes Aircraft Company
P.O. Box 3310
Fullerton, CA. 92634
(714) 441-9694

ABSTRACT

A trusted computer system consists of an array of software and hardware mechanisms that implement and enforce the system's security policy. These mechanisms must therefore resist tampering by adversaries who wish to modify or replace specific elements to facilitate attacks against the system. Software integrity can be verified with cryptographic checksums, but techniques for verifying the integrity of hardware elements to the same high level of confidence do not exist. Even the recently released *Integrity In Automated Information Systems* supplement fails to adequately address the problems of verifying systems integrity at the hardware level. Physical protection is therefore commonly used to restrict and/or detect access to the equipment, and the *Guide To Trusted Distribution* suggests a variety of techniques for implementing such mechanisms. However, they may not be sufficient to prevent a motivated adversary from successfully attacking critical elements during distribution between sites or during operational use.

This paper presents another method for detecting unauthorized access that offers a greater level of protection than the methods currently available. By combining random number generation and asymmetric encryption, an electronic seal can be implemented that cannot be forged, bypassed, or replaced. Even legitimate parties will not be able to access the protected equipment without leaving evidence.

Keywords: *Continuous protection, hardware integrity, trusted distribution, trusted facility management, configuration management.*

INTRODUCTION

One of the features required by the *Trusted Computer Security Evaluation Criteria* (TCSEC) at Division A is trusted distribution of all software and hardware elements[1] of the Trusted Computing Base [1]. The intent is to prevent an adversary from undetectably tampering with the system or its components during shipment from one site to another, or from replacing them with counterfeit versions that may contain hostile functions. Software

[1] The term "hardware element" is used to represent individual components (e.g., integrated circuits), circuit boards (e.g., video or disk drive controllers, etc.), and subsystems (e.g., video monitors, tape backup systems, etc.).

tampering can be easily detected with cryptographic checksums at any time during the system's life. Achieving an equivalent level of confidence for the hardware is a much more difficult problem, however. While every byte of a computer program or data file can be observed and validated, most of the gates and functions within a hardware element simply cannot be observed at a level of granularity that permits verification that the current implementation is identical to the desired implementation. Modern built-in-test and fault isolation techniques may not even detect elements that had been replaced with malicious versions. (An adversary would most likely design the hostile element such that it would continue to provide the response expected by the system.) Recognizing these fundamental limitations, tamper detection methods that can be applied to shipping containers, and to the system's housings and enclosures[2] during operational use, are suggested by the *Trusted Distribution* supplement of the TCSEC [2] as a partial solution.

One such method is an active system that generates an alarm when the enclosure is opened. The typical problem with these systems is that the intended recipient must be able to reset the mechanism once triggered or after performing routine maintenance. An adversary may therefore also use this capability to reset the mechanism after replacing or modifying its contents. A related device generates and displays a random number each time the enclosure is opened. However, these devices can usually be replaced with a forged unit, programmed to display the original number, to conceal the fact that an intrusion had taken place. This paper describes an active tamper detection system that eliminates these vulnerabilities. It can be used to help meet the trusted distribution requirement as well as to help satisfy life-cycle integrity assurance requirements.

[2] The term "enclosure" is used to represent shipping containers, equipment housings, and in general the physical barriers of the environment to be protected.

Background

The techniques suggested in *Trusted Distribution* were evaluated to determine the most effective method for detecting unauthorized access to hardware elements during storage, distribution, and in operational use. The adversary was assumed to have sufficient motive, capability, and opportunity to modify or replace any hardware element he wished to target. Given this worst-case scenario, it was determined that the suggested techniques may not be able to adequately protect a high-value system against such an adversary.

The methods of protective packaging, couriers, registered mail, message authentication codes, encryption, and site validation were suggested as possible solutions. Since each application and its threats are unique, the vendor is allowed to submit a plan describing his specific approach. The plan may be based on the suggested methods, on variations of them, or on completely different techniques. It is then included in the overall evaluation process.

One protective packaging technique is to shrink wrap the hardware, possibly with some type of unique or specially treated material. The main disadvantage of this approach is that the material may be obtained through its manufacturer, other vendors, or even stolen from the original equipment's vendor, thus allowing an adversary to rewrap the equipment without detection. It also cannot be applied to physically large equipment, and it cannot be used once the system is installed and operational.

Another protective packaging technique is an active monitoring system, such as an audible or visual alarm, but they suffer from the previously mentioned problem that they can usually be turned off or reset.

The last protective packaging method is the tamper-resistant seal. Paper seals that cannot be removed without their destruction are the most common, and are usually applied over two mating surfaces or over the screws that hold cover panels in place. (A wire and wax seal placed through a locking mechanism is a related technique.) A validation stamp is also

often used to identify the final inspector or person who applied the seal. These methods offer only minimal protection since the seals can usually be easily obtained or copied, and validation stamps can be easily forged. Personnel authorized to apply the seals can also be bribed or extorted into supplying the adversary with new seals. Holographic seals are substantially more difficult to copy, but they can still be obtained from converted personnel. Electronic seals are also available, such as devices that generate random numbers (or record the date and time) whenever covers are removed or containers are opened. The concept is that every access will cause a different random number to be displayed. Tampering can then be detected by comparing the current number with its value prior to shipping. The main weakness of this approach is that an adversary can simply remove the device and replace it with a similar one that displays the original number. The false device may even continue to generate random numbers with successive triggers to avoid drawing suspicion to itself.

The use of a courier service is also suggested. Its advantage is that the equipment will be under constant surveillance by bonded or otherwise trusted individuals. However, even trusted persons can be converted by bribery or extortion, and its cost is significantly higher than methods that do not require continuous personal attention.

Registered mail, supplemented by methods that ensure the identity of the sender, is also suggested. As with couriers, however, personnel can be converted. In addition, constant surveillance is not provided, allowing an adversary the opportunity to access the equipment.

Message authentication codes, encryption, and checksum programs are suggested, but are only suitable for use on software.

Lastly, on-site validation is suggested as an additional layer of assurance that the hardware had not been tampered with during its distribution. Physical inspections are discussed, but inventory inspections can only detect gross discrepancies, and engineering inspection by qualified technicians may not detect attack by a competent adversary who conceals his efforts.

A TAMPER-RESISTANT ELECTRONIC SEAL

Since none of the above methods were considered sufficiently robust to withstand the efforts of a motivated adversary, the Tamper Detection System (TDS) was developed[3]. When implemented with an array of suitable sensors, the TDS provides a cost-effective, reusable seal that cannot be forged, bypassed, or replaced. In fact, even persons authorized to access the equipment will not be able to do so without leaving evidence.

A Display Module and an array of tamper sensors are placed within the area to be protected. When any sensor is activated, the Display Module generates a random number, referred to as the audit count (Figure 1). The audit count is then encrypted and stored, and a sequence number is incremented.

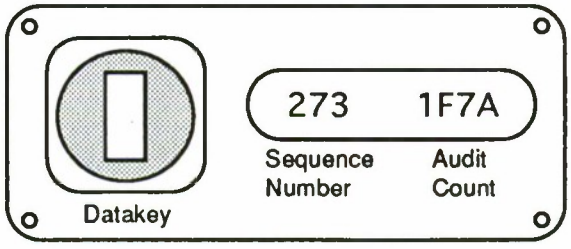


FIGURE 1. The Front Panel of the Display Module.

To later recover this information, a trusted individual inserts a Datakey[4] containing the corresponding decrypt key. The decrypt key is extracted, and the encrypted audit count is decrypted and displayed along with the sequence number. Verification that the sequence number and audit count had not changed since the last time they were examined provides assurance that access had not been gained. The combined operations of generating a random number each time access

---

[3] Patents on this device are pending in the U.S. Patent Office.

[4] Datakey is a trademark of Datakey, Inc., Burnsville, MN.

takes place, and limiting its recovery only to authorized personnel in possession of a uniquely keyed and physically separate device, results in an unforgeable seal. This provides its users with a very high level of confidence that the enclosure's integrity had not been breached.

Functional Description

A functional block diagram of the TDS is shown in Figure 2. The Display Module contains all the required circuitry and the Datakey interface. It is installed within the monitored enclosure, but with its display and Datakey receptacle externally accessible. The Datakey is a physically separate device that is assigned to the trusted individual responsible for maintaining the security of the system (e.g., the System Security Officer, or SSO).

The Algorithm element performs the encryption and decryption functions. A public-key algorithm is used so that compromise of the Datakey will not compromise the device's effectiveness [3]. (The Data Encryption Standard could also have been used, but loss of its common encrypt and decrypt key could allow an adversary to create a false Display Module.)

The Random Number Generator element produces the random numbers used as the audit count. A hardware randomizer, based on a random physical process to ensure a non-deterministic output (e.g., thermal noise from a semiconductor junction), is used to prevent an adversary from predicting the next (or previous) audit count. A software-based pseudorandom number generator could also have been used to reduce the cost of the TDS, but it may not resist attack as well as a noise-based randomizer.

The NV-RAM element provides non-volatile storage of the Display Module's serial number and encrypt key, and the current sequence number and encrypted audit count. An electrically-erasable PROM (EEPROM) is used since it retains its contents even if power is lost or disconnected.

The Display element is the visual display for indicating the sequence number and audit count. Three decimal characters are used for the sequence number, and four hexadecimal characters for the audit count. This supports 1,000 sequence numbers and 65,536 different audit counts.

The Sensors element interfaces with normally-open and normally-closed sensors. Any type of tamper sensor can therefore be used, since even the most sophisticated ones are compatible with these standards. When triggered, they initiate the generation of a new sequence number and audit count.

The Control element directs the various internal operations. It also allows programming of the serial number and encrypt key when a special Initialization Datakey is inserted into the Display Module.

The Sequence Counter element reads the current sequence number from NV-RAM, increments it by one, and stores it back in NV-RAM. The sequence number is not encrypted since this would require that the decrypt key be retained within NV-RAM, reducing the work factor required for an adversary to recover it. The overall security of the system is not reduced as long as the audit count remains encrypted.

The Power Control element performs two functions. First, it removes power from all elements except Sensors to minimize power consumption. When a sensor is triggered, it turns the other elements on so they can perform their functions. Power is then removed until the next activation. Its other function is to monitor the input power to detect a low voltage condition.

The serial number allows a single Datakey to store the decrypt keys of several Display Modules. This eliminates the need for an SSO to manage multiple Datakeys.

When Sensors is activated, Control extracts the encrypt key from NV-RAM and transfers it to Algorithm. The Random Number Generator then generates a new audit count, which is encrypted by Algorithm and stored in NV-RAM. The unencrypted audit count is then purged from memory. The sequence number is also incremented and stored.

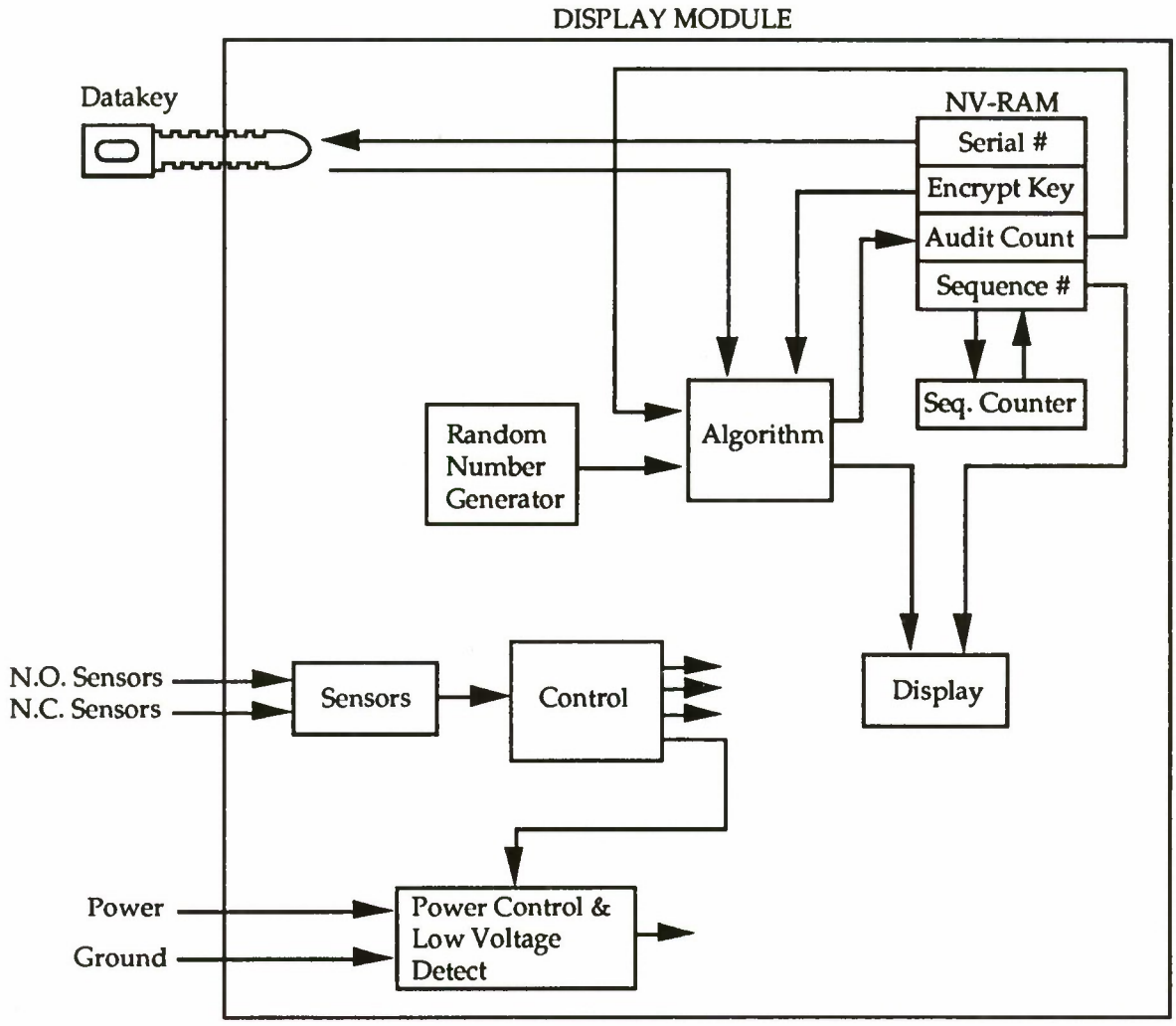


FIGURE 2. Functional Block Diagram of the TDS.

When the Datakey is later inserted, Control extracts the decrypt key and transfers it to Algorithm. The encrypted audit count is then read from NV-RAM and decrypted. The sequence number is also extracted, and both are displayed until the Datakey is removed. If these values do not agree with their previously noted values, then an unauthorized intrusion had taken place.

Implementation

A block diagram of a representative implementation is shown in Figure 3. Using commercially available components, this circuitry can be contained entirely within the Display Module itself (with the exception of the sensors and an optional external power source).

A low-power microcontroller continuously scans the tamper sensors for activity and for insertion of a Datakey. When triggered, it energizes the other circuits until all processing is successfully completed, then shuts them down to conserve power.

A hardware randomizer produces the audit counts in this implementation, but it can be eliminated if a firmware-based pseudorandom sequence generator is implemented within the microcontroller.

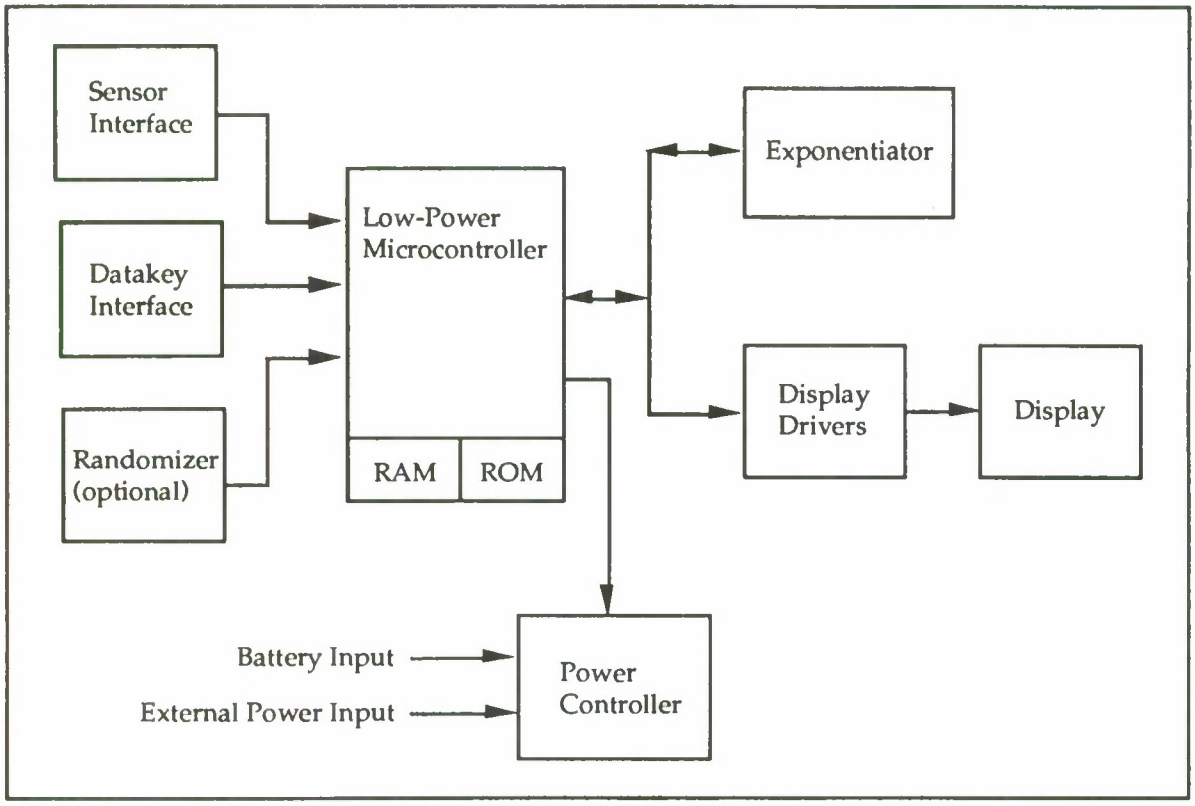


FIGURE 3. A Typical Implementation of the TDS.

A hardware exponentiation device, such as those available from the Cylink Corporation, performs the encryption and decryption mathematics. (Performing these calculations in firmware would take too long to execute.)

Liquid crystal displays are preferred due to their lower power consumption, but other displays with appropriate drivers can be used as well.

A TRUSTED DISTRIBUTION APPLICATION

For a trusted distribution application, the Display Module, and its sensors and batteries, must be installed within the desired shipping container. (Since the container must not allow access to its contents without triggering one or more sensors, it must be specifically designed to ensure that they cannot be bypassed.) Any commercially available sensor can be used, such as microswitches, pressure switches, or light detectors. Most are compatible with each other, so they can even be used in combination. However, it is the selection and installation of these sensors that usually proves to be the most difficult aspect of the implementation. Ensuring that they always trigger when access is gained, never trigger falsely, and cannot be bypassed can be tricky.

Once the container is filled and ready for shipping, the Display Module must be programmed with its serial number and encrypt key. The SSO therefore uses the Key Generation Support System (KGSS), a special-purpose computer program hosted on a MS-DOS personal computer. After entering the desired serial number, a matching encrypt-decrypt key pair will be generated. (A database of previously generated key pairs could also have been used.) The KGSS then loads the serial number and encrypt key into an Initialization Datakey through its serial port. Likewise, the serial number and decrypt key are loaded into an Operational Datakey for everyday use by

the SSO[5]. Obviously, access to this program must be carefully controlled to prevent compromise or malicious modification (such as to force the generation of key pairs of the adversary's choosing). This can be accomplished with conventional trusted mechanisms or a dedicated computer under strict physical control.

Figure 4 illustrates the operation of the Display Module. The Initialization Datakey is first inserted into the Display Module. When the Initialization Datakey is recognized, its serial number and encrypt key are extracted and stored. The sequence number is incremented, and a new audit count is generated, encrypted, and stored. (The unencrypted audit count is overwritten from memory.) The Initialization Datakey is then removed and the Operational Datakey, containing only the serial number and decrypt key, is inserted. The sequence number and audit count will then be displayed, and can be recorded in an audit log. The Operational Datakey is then removed. The Initialization Datakey can then be erased by the KGSS for reuse at a later time.

The container is now sealed and it can be released for shipping. The Datakey and audit log are sent to the recipient by independent means (e.g., secure phone call, Registered Mail, etc.). Upon arrival, the local SSO inserts the Datakey and verifies that the sequence number and audit count were identical to those supplied in the log. Since any attempt to access the container will result in a new sequence number and audit count, the recipient can have a high degree of confidence that the container held legitimate and unaltered hardware (or software) elements. In addition, the container itself could not have been surreptitiously replaced since an adversary cannot determine the pre-attack values in order to program a forged TDS.

[5] The same type of Datakey is used in both applications. Parameter fields are used to distinguish Initialization Datakeys from Operational Datakeys.

A LIFE-CYCLE INTEGRITY ASSURANCE APPLICATION

The same process can be used to monitor a computer system for unauthorized internal access. The Display Module is mounted within the computer with its display visible through a small window, and a number of microswitches or other types of sensors are installed on the various cover panels. A rechargeable battery supplies power when line power is not available.

After initial system installation, the SSO inserts the Datakey and records the date, time, sequence number, and audit count in the system log, then removes the Datakey. This is repeated periodically, perhaps at the beginning of each shift. If the sequence number or audit count were ever different from the last log entry, with no justification, a breach of the system's integrity may have occurred. Appropriate steps could then be taken to investigate the cause.

Occasionally, routine maintenance will need to be performed on the computer, forcing generation of a new sequence number and audit count. This situation is easily handled. The SSO inserts the Datakey and records the new information in the log, along with an explanation of the maintenance performed and the name of the technician. The technician should also verify the entry and sign the log. This provides an audit trail of who had internal access to the computer, when it took place, and for what reason.

SECURITY FEATURES

In order for an unauthorized access to go undetected, the sequence number and audit count must be returned to their previous values. Without knowledge of the decrypt key, however, an adversary must recover the encrypt key, algorithm, and encrypted audit count from the Display Module, and then use them to attempt to calculate the decrypt key. This requires that significant reverse-engineering and computational resources be applied to that particular Display Module, and would therefore be too expensive, time consuming, and risky to be practical.

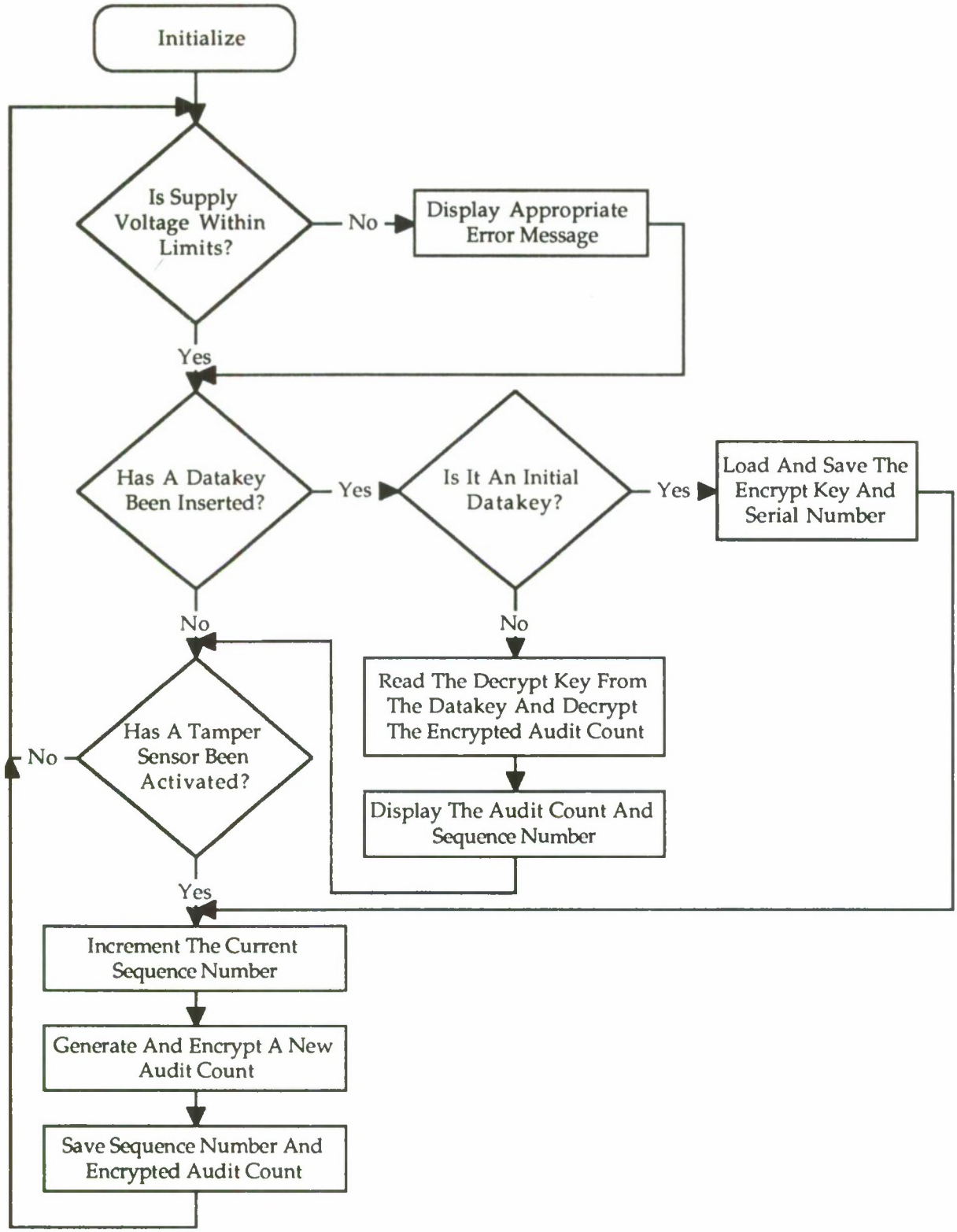


FIGURE 4. Flowchart of the Display Module's Operation.

If the audit count were not stored in encrypted form, then it could be recovered by probing the Display Module's memory (assuming that it could be accessed without triggering any sensors). However, the computational complexity of cryptanalysis with only a small amount of ciphertext increases the work factor required for the attack to succeed to the point of being impractical.

The use of random audit counts prevents undetected replacement of the Display Module or forgery of the system logs. For example, if a simple incrementing access count was used, anyone could read the last count from the log, make a new entry with it incremented by one, and forge the SSO's signature. Possession of the Datakey would not be required for this attack to succeed.

If a sequence number were not used, it would be theoretically possible, although not very likely, that an adversary in possession of the Datakey could repetitively remove and replace a cover until the previous audit count reoccurred. Use of a sequence number therefore ensures that this type of attack will not succeed because the resulting gap in sequence numbers will be obvious. Eventually, however, the sequence number will reach its maximum and begin counting again from zero. Since the audit counts are random, though, the probability of the same audit count occurring when the sequence number reached its previous value is extremely small. In any case, this could require a significant amount of time, thus increasing the adversary's risk of being discovered.

Replacing the encrypt and decrypt keys or the serial number will also not allow subversion of the TDS's protection. This simply initiates the same processes as if a tamper sensor had been activated. Since the sequence number is incremented, not reset, and since a new audit count is generated, encrypted, and stored, this attack will not succeed. The different sequence number and audit count displayed to the SSO the next time he checks the system will indicate that it had been tampered with.

The Display Module cannot be replaced with a false unit because the current audit count cannot be determined by the adversary, and power cannot be removed after access is gained in time to interrupt the process. For example, a simple microprocessor-based implementation with a hardware exponentiation chip can complete all of these operations within 250 milliseconds. Clearly, this does not allow enough time for a cover to be removed and its power to be interrupted before a new audit count can be generated and stored.

Possession or access to the KGSS will also not allow the TDS to be subverted. Once a matching encrypt-decrypt key pair is generated and loaded into a specific Display Module and its Datakeys, that key pair is longer needed by the KGSS and is therefore purged from memory. However, it must be expected that someone will eventually lose their Datakey. At that time, a new encrypt-decrypt key pair can simply be generated and loaded into the system. This slight inconvenience is offset by the advantages of less complex handling procedures for the KGSS and the significantly reduced possibility of compromise of fielded key pairs.

Although it is still possible for the "trusted" person with the Datakey to gain unauthorized access and update the log with the new audit count and a false reason, attacks by anyone else will be detectable. Since this places a greater degree of responsibility on the persons who possess the Datakey, and limits the number of suspects if an unauthorized modification is later discovered, a deterrent effect is provided as well.

LIMITATIONS OF THE TDS

Although the TDS eliminates many of the vulnerabilities typical of more conventional approaches, it cannot eliminate them all.

For example, persons in possession of the Datakey, or those with access to the KGSS, can still make unauthorized modifications to the system and log the event under a false pretense. The same is true of persons who have the opportunity to copy someone else's Datakey. Implementing the "two man control" rule by requiring two Datakeys to be inserted at the same time is one solution to this problem.

False alarms, and failing to detect intrusions due to intermittent or faulty sensors, are also as much of a problem for the TDS as they are for

any of the conventional approaches. The reliability of modern electronics is very good; it is the selection and installation of the sensors, and the mechanical design of the enclosure, that are usually the Achilles heel of such systems. They must provide adequate coverage over the environment to be protected, and they must remain immune to external influences. For example, the operation of resistive pressure sensors can be influenced by temperature variations of natural or hostile origin. The same is true of magnetic reed switches with respect to externally-applied magnetic fields. Improper selection or placement of sensors may therefore allow internal access to be gained without detection, at which point the sensors may be electrically or physically bypassed to facilitate future attacks.

CONCLUSION

Although trusted distribution is not required until Class A1, life-cycle integrity assurance is required of all systems at Class C1 and higher. While the *Integrity In Automated Information Systems* supplement of the TCSEC [4] attempts to define the properties of integrity, it stresses mechanisms applicable primarily to software. It is therefore hoped that the methods presented here will prove useful to those developing trusted systems and to those responsible for maintaining their secure operation.

It should be clear that while passive and active tamper detection systems can offer varying levels of protection, they must be designed carefully if they are to resist the attacks of a motivated and skilled adversary.

It must be remembered, however, that detecting unauthorized equipment access only addresses a small part of the much larger problem of hardware authenticity. Hostile attacks at the integrated circuit level can affect data integrity and system availability, and attacks at the card or subsystem level can easily compromise data confidentiality as well. Knowing that an adversary had accessed these elements does not help us to determine what, if anything, may have been done to them. What is really needed are mechanisms to provide the same high level of assurance for hardware integrity as the cryptographic checksum now provides for software. As trusted systems find use in an ever increasing variety of applications, we must search for robust, effective, and economical methods to ensure that unauthorized hardware or software modifications cannot go undetected.

REFERENCES

[1] Department of Defense, *Department of Defense Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, December 1985

[2] National Computer Security Center, *A Guide To Understanding Trusted Distribution In Trusted Systems*, NCSC-TG-008, 15 December 1988

[3] Meyer, Carl H. and Matyas, Stephen M., *Cryptography: A New Dimension In Computer Data Security*, John Wiley and Sons, 1982

[4] National Computer Security Center, *Integrity In Automated Information Systems*, C Technical Report 79-91, September 1991

A TCB SUBSET FOR INTEGRITY AND ROLE-BASED ACCESS CONTROL *

Daniel F. Sterne
Trusted Information Systems, Inc.
Glenwood, Maryland

Abstract

The Extended Access Control Subsystem (ECS) is an experimental prototype providing a set of configurable access controls. These controls are intended to support a range of application-specific security policies. The design of ECS represents a synthesis of important ideas from the research literature including non-hierarchical domains, the Clark-Wilson integrity model, role-based policies, object-state-based controls, and use of hardware protection rings. ECS has been designed as an untrusted TCB subset for use with an extended version of the Trusted XENIX[1] operating system. The features of ECS are described, and the impact of the TCB subset approach on its design is discussed.

1 Introduction

The evolution of trusted system principles and technology has historically been oriented toward addressing the DoD confidentiality problem. There is increasing recognition, however, that systems are needed that can be trusted with respect to other kinds of security concerns, especially those of the civil sector. Building these systems confronts a number of open issues, including the following:

- What security policies other than DoD confidentiality should be supported by automated mechanisms, and what mechanisms are appropriate ?
- What trust principles and criteria should guide development and evaluation ?
- What engineering strategies facilitate timely and cost-effective development and evaluation ?
- How can such systems achieve a reasonable degree of compatibility with existing products and applications programs ?

This paper describes part of a research project that attempts to explore these issues. The current focus of the project is the construction of the Extended Access Control Subsystem or "ECS". ECS is a research prototype that provides a collection of access control mechanisms that are configurable to support a range of application-specific security policies. The design of ECS represents a synthesis of key ideas taken from the research literature; these include non-hierarchical domains, the Clark-Wilson integrity model, role-based policies, object-state-based controls, and use of hardware protection rings. ECS has been designed to be an untrusted TCB subset [30] that will operate in conjunction with an extended version of Trusted XENIX [9], a trusted operating system evaluated by NCSC at the TCSEC B2 evaluation class. ECS is the first example

*Funded by DARPA through RADC contract F3062-89-C-0125.
[1]XENIX is a registered trademark of the Microsoft Corporation.

known to the author of applying TCB subsetting to the enforcement of policies other than mandatory access control (MAC) or discretionary access control (DAC). As such, it provides an opportunity to explore the applicability of TCSEC-oriented principles and evaluation criteria to the construction of systems that are trusted with respect to other security policies.

The remainder of this paper is organized in the following manner. Background on ECS and TCB subsetting are provided by Sections 2 and 3. Section 4 describes ECS features and requirements. Section 5 describes the high-level architecture of ECS. The influence of TCB subset considerations on the design of ECS are explained in Section 6. Section 7 provides a summary.

2 Background and Rationale

ECS requirements and features were chosen after a survey of DoD and civil-sector security policies [32]. ECS is primarily intended to support "organizationally-directed" policies, sometimes referred to as non-discretionary policies, in which an organization dictates a fixed division of authority for resource usage. Computer systems used in this context are typically characterized as turn-key or embedded systems. Such systems typically do not allow system users to extend the system's software or create personal files. By contrast, general purpose computing systems allow users to create and modify programs and other files. They also allow users to give other users access to these files.

The key features of ECS support role-based access, the Clark-Wilson model of integrity, flexible non-hierarchical domains, and object-state-based controls. These features, described in the next section, were selected based on perceived applicability, flexibility, and potential for relatively high assurance implementation. In addition, because of resource limitations, the project emphasized extending ideas previously studied by other researchers, and deemphasized inventing entirely new ones. Resource limitations also encouraged a design strategy based on the notion of TCB subsets. Using this strategy, ECS has been designed as an untrusted layer added to the Trusted XENIX operating system. In principle, ECS could be added, removed, or modified without affecting the trustedness of a Trusted XENIX configuration.[2]

This strategy has reduced ECS development time by allowing ECS to make use of existing Trusted XENIX features and assurances. For example, Trusted XENIX provides identification and authentication, audit, trusted path, subject and object abstractions, and non-bypassable tamper-resistant access controls. Moreover, since TCB subsetting is a "non-invasive" approach, the integrity of the Trusted XENIX TCB is protected from adverse interactions with ECS. This potentially reduces the scope of ECS debugging efforts.

ECS is being developed consistent with B2 architectural assurance requirements, including least privilege, modularity, and effective use of hardware. The high-level design of ECS is nearly complete, and has been supported by detailed prototyping of the kernel extensions discussed in Section 5 [24]. Approximately half of the anticipated ECS code modules have been implemented and are beginning initial integration tests. The coding of a demonstration application is also underway.

3 TCB Subsets

The notion of a TCB subset was initially described by Shockley and Schell [30], and has more recently served as the basis for the Trusted Database Management System Interpretation (TDI) of the TCSEC [34]. A TCB subset is a generalization of the reference monitor abstraction. The primary difference is that "a TCB subset may have an internal interface to a smaller included mechanism, which is also a TCB subset" [30].

[2] In practice, the Trusted XENIX kernel must first be modified slightly to allow this. These modifications are discussed in subsequent sections.

This notion provides a strategy for designing and evaluating complex TCBs as collections of simpler TCB subsets. Each TCB subset enforces a subset of the system access control policy, described in [30] as "the rules concerning the access of subjects to objects". Motivations for this strategy include reducing TCB complexity, reducing TCSEC evaluation effort, and reducing development and evaluation cost and risk for vendors wanting to extend the security perimeter of an existing TCB. The TCB subset strategy imposes a number of requirements on the design of a TCB. Although a thorough treatment of these is beyond the scope of this paper, a few key requirements must be cited to provide context for the discussions that follow.

A TCB subset, like a reference monitor must be non-bypassable, tamperproof, and designed to allow complete analysis and testing. Consequently, every TCB subset responsible for protecting a particular object must be consulted on every access to that object. Moreover, in combination, the selected subset policies must be at least as restrictive as the desired system policy.

Each subset must be allocated to a set of protection domains that are partially ordered by privilege.[3] TCB subsets may be hierarchically ordered such that the domain of one subset includes that of other subsets and is hence more privileged. A less privileged subset must have no capability to bypass or tamper with a more privileged subset, and must be *untrusted* with respect to the policy enforced by it. In principle, if these conditions are met, it should be possible to evaluate a less privileged subset largely independently of previous successful evaluations of more privileged subsets.

The two subsets discussed in this paper, ECS and an extended version of Trusted XENIX, are hierarchically ordered, and have been designed in accordance with TCB subset requirements.

4 ECS Features

4.1 Non-Hierarchical Label-Based Domains

The foundation of ECS is a set of software facilities to define and enforce an application-specific configuration of execution domains. (The relationship between domain configurations and security policy enforcement is explored in [28].) These facilities, which are similar to those provided by the LOCK system [5, 35, 23], restrict the ability of subjects to access objects based on domain definitions that are described in terms of data types. ECS associates a special unchangeable security attribute with each Trusted XENIX object that indicates its type. This attribute is sometimes referred to as a type label because of its similarity to a MAC label. Nevertheless, type labels are independent of the MAC labels that Trusted XENIX associates with objects. Except for a few types that ECS reserves for marking and protecting its own access control tables and other metadata, a particular type has no inherent meaning to ECS. Types are simply a means for applications designers to identify equivalence classes that are used in defining domains.

Every ECS subject executes an ECS domain. Domains are specified when ECS is configured for a given system installation. Each domain specification lists all of the types of data that are accessible to a subject in that domain and, for each type, the permitted modes of access (i.e., read or read/write). In this way, the input sources and output sinks for each subject can be precisely controlled, and information flow among system components can be configured to suit the application [35, 36, 23]. While ECS allows an arbitrary configuration of domains to be established, domains cannot be changed or created "on-the-fly", i.e., while the system is in operation. Domain configuration changes are possible only when ECS is in a special maintenance mode under control of the Trusted XENIX Trusted Systems Programmer.

Unlike LOCK, which uses a special purpose security coprocessor called the SIDEARM [23] for type and domain enforcement, ECS relies only on software and a general purpose CPU.

[3] The partially ordered domains referred to here should not be confused with the unordered ("non-hierarchical") domains discussed in the next section. The former, which are enforced by hardware mechanisms, are exported to ECS by Trusted XENIX, while the latter are exported by ECS to applications.

4.2 Support For the Clark-Wilson Model

One of the key ideas advanced by the Clark-Wilson integrity model [6, 7] is that controlled operational data must only be modified by programs that have been certified to change the data in a constrained and appropriate manner. These programs are referred to as transformation procedures or TPs. Objects holding operational data are referred to as Constrained Data Items or CDIs. Rules C2 and E1 of the model describe this requirement by referring to sets of access relations of the form [TPi, (CDIa, CDIb, CDIc,...)]. Each relation stipulates that a particular TP (program) can be permitted to access a particular set of CDIs. The model also states that relations may use wild cards to match classes of CDIs.

The domain facilities provided by ECS have been specifically tailored to support an extended version of these relations. In ECS parlance, each extended relation is referred to as a "bound operation definition". This term is intended to suggest the "binding" [6, 4] of a TP to a set of CDIs, or, alternatively, the binding of an operation (program) to particular data types. Bound operation definitions extend Clark-Wilson relations in three ways:

1. A bound operation definition refers to a named TP and a set of *CDI data types* rather than a set of specific CDIs.
2. A bound operation definition also includes access mode constraints to be enforced on the TP. For each CDI data type listed, the set of permissible access modes is specified.
3. A bound operation definition includes a list of other bound operations that may be invoked or signalled.

This approach is borrowed from Thomsen's work [35, 36] on the LOCK project, which describes the rationale and many additional details. The essence of the approach is that each TP should be externally constrained so that it can access only the CDIs necessary to its purpose, and only in necessary access modes. This is in accordance with the principle of least privilege and is motivated in part by an assertion by Clark and Wilson [6]:

> ... an important research goal must be to shift as much of the security burden as possible from certification to enforcement.

Access constraints on TPs are implemented by executing TPs as distinct subjects in highly-restricted non-hierarchical domains. In Thomsen's work, each domain is specified by a row in the domain definition table. TPs and relations, however, are not explicitly represented, because a single domain may be shared by multiple TPs and relations. Any object whose type is executable in the domain may represent a TP, or a part of a TP. The relation (or collection of relations) associated with the table row is implied by the row's elements; these identify the types that are executable, readable, and writable in the associated domain.

By contrast, ECS data structures are list-oriented rather than tabular, and have been tailored to provide a more explicit representation of TPs and extended Clark–Wilson relations. In ECS, lists of bound operation definitions are used in place of domain definition and domain transition tables. A separate bound operation is specified for each desired relation. Furthermore, all TPs belong to a single type. Consequently, a bound operation definition refers to its TP by program name rather than by type, as indicated in item 1 above. This has the effect of creating a unique execution domain for each relation. The domain includes execute access to the TP, and read or read/write access to specified types, as shown in item 2 above.

Each ECS subject is associated with a single bound operation. When a subject is created, it begins executing the corresponding TP. Nevertheless, an ECS subject can call or interpret other programs (i.e., without a domain change). Logically, such programs are parts of the TP that have been packaged as separate executables for reusability, size, or other reasons. An additional ECS feature allows each bound operation definition to denote one of its writable data types as being the default type. This type is automatically

assumed at run time when a TP attempts to create an untyped output object. These features provide ECS-compatibility for existing executables that call other executables and create objects without referring to type or bound operation names. For example, they allow a command line interpreter such as a Unix[4] shell to be installed as a TP. Compatibility with existing software is discussed further in a later section.

A bound operation is also permitted to invoke other bound operations identified as invocable in its definition, as indicated in item 3 above. Because each bound operation executes in its own domain, invoking a bound operation is analogous to executing a LOCK domain transition. ECS provides the *invoke bound operation* service for this purpose. A successful call to this service terminates the calling subject and creates a new subject that is confined to a different domain. As discussed in later sections, the new subject is created within the process formerly occupied by the caller.

4.3 Role-Based Constraints

Abstractly, a role is a set of rights to access, operate on, or otherwise use resources in particular ways [14, 18, 2, 35]. In ECS, a role provides the right to execute a collection of bound operations arranged in a tree-like structure. This structure is formed by the invocation and signalling linkages that are part of bound operation definitions (see item 3 above). This corresponds to a collection of LOCK domains having inter-domain links represented by entries in a LOCK domain transition table [5, 35].

In ECS, a role definition identifies the "start-up" bound operation that is to be initiated at login time when the role is assumed. Typically, the start-up bound operation will provide a menu of commands or act as a restricted command line interpreter. The start-up bound operation is the root of the role's invocation tree; the bound operations that are this bound operation's descendants in the tree (if any) are those that are listed in its definition as being accessible via invocation and signalling. The definitions of these descendant bound operations similarly identify their own descendants, and so on. A bound operation that is the root of one role tree may be a descendant node of the root of another role tree. In this way, new roles can be built by referring to existing roles, and hierarchical relationships [2] between roles can be directly represented.

Implementing a role as a collection of execution domains allows an authorized *user* to be given simultaneous read and write access to an arbitrary collection of data types, while confining each *TP* accessible to the user to a small, highly restrictive domain; this facilitates use of fine–grained least privilege [35]. In this way, a single user session can be given access to a collection of execution domains with precisely defined and controlled domain interactions.

Associated with each ECS role or group of roles is a user authorization list. This list identifies the users who are authorized to assume the role at login time. In addition, the role may have other constraints associated with it that limit the ability of users to be authorized for the role or to activate the role. For example, some roles may be accessible only if not simultaneously in use by other individuals [14, 18], or only during login sessions at a particular MAC level [18]. These constraints are enforced by the predefined ECS Login Program. Static separation of duty constraints [6, 20] can also be specified; these prohibit a single individual from being authorized for a combination of roles that provides access to *critical combinations* of bound operations. Static separation of duty constraints are enforced by the User Authorization Program, a predefined TP used to add users to authorization lists.

In ECS, one role can be given custody over the user authorization list for another role. This allows supervisory roles to dynamically authorize individuals to fill subordinate roles under their supervision. If a subordinate role includes a subset of a supervisory role's bound operations, a form of delegation of authority is provided. Distribution of role custody can be implemented in ECS without use of additional mechanisms. This is accomplished by assigning different types to the authorization lists for different roles. By binding the User Authorization TP separately to each of these types, a collection of role-authorizing bound operations is created. Each bound operation can then be included in the invocation tree of the appropriate supervisory role.

[4] Unix is a registered trademark of AT&T.

4.4 Object-State-Based Controls

One of the ideas that emerged in papers stimulated by the Clark-Wilson model is that of dynamic separation of duties [21, 13, 26]. This form of separation of duties constrains access to an object based on its previous access history. The historical information of interest includes the identities of individual users who have accessed it and the TPs they used. In the financial control example described in [26], the processing of a check requires three sequential steps: prepare, approve, and issue. Although some individuals may be authorized to perform more than one of these steps on checks in general, no individual may perform more than one step on any particular check.

The object-state-based controls provided by ECS are derived from those described by Sandhu [26] for enforcing dynamic separation of duties. In ECS, these controls provide an additional layer of constraints that can be selectively added to the domain controls discussed above. In Sandhu's approach, a "transaction control expression" is associated with *each type* of controlled object. The transaction control expression describes the sequence of TPs (transactions[5]) that can be applied to each object and the associated separation of duty constraints, e.g., no individual may apply more than one TP to any object. Associated with *each object* is a "mini-audit log" that is protected and updated by the TCB. This log captures the object's access history in terms of the user and TP identifiers. Each time an object access is attempted, the mini-audit log and the transaction control expression are consulted. If the separation of duty constraint is not met, the attempt is rejected; if the attempt is successful, the log is updated accordingly.

In ECS, TP sequencing controls are specified as state transition tables that are equivalent to finite state automata. These controls can be used with or without separation of duty constraints, which rely on user identifiers. This allows sequencing controls to be used in systems in which identification and authorization are handled procedurally, and user identifiers are not maintained by the system (e.g., Aegis [32]).

ECS state transition tables consist of four columns. The columns represent a current state, a TP identifier, new state, and an optional separation of duty specifier. The latter is used to mark combinations of transitions that cannot be (or must be) executed by the same individual. Each row describes a permissible state transition from a given current state to a new state, whereby the transition is accomplished by applying the specified TP. Attempts to access an object whose type has an associated state transition table will be rejected unless the attempt is consistent with the sequencing and optional separation of duty constraints described in the table.

During ECS system configuration, object-state-based controls for sequencing or dynamic separation of duty can be established optionally for any type. This is accomplished by attaching a state transition table to the type's definition. For each object belonging to such a type, ECS will create and maintain a current state indicator, and if necessary, a mini-audit log. These constitute dynamic metadata that cannot be modified directly by a TP; TPs can only affect these indirectly as a result of accessing the objects whose states they describe. Unlike other types of objects that can be accessed using existing Trusted XENIX system services (e.g., open, read, write), objects of state-controlled types can only be accessed via specialized ECS–provided services. These services integrate write access, mutual exclusion, and synchronized updating of current state indicators and mini audit logs.

4.5 Orthogonality and Compatibility With Trusted XENIX

ECS provides a set of access controls that are logically independent of MAC and DAC. Access control requirements of different applications may vary. Hence, it is desirable that a configured system be able to make use of either ECS, or MAC and DAC, or some combination thereof [14, 18]. A TCB or TCB subset, however, must be non-bypassable. As a result, selective use of one kind of access controls must be accomplished by configuring all others so that their associated access checks always succeed.

[5] Sandhu assumes that TPs are full-fledged transactions exhibiting serializability and failure atomicity, properties not assumed here.

If desired, ECS can be configured so that its checks always succeed. All objects can be assigned to a single data type, and all subjects to a single execution domain in which the sole data type is readable[6] and writable, and is the default output data type. Only a few bound operations are needed (apart from predefined bound operations that are used to configure ECS). Each simply binds this single domain with a command interpreter TP that is presented to a user at the beginning of a session. The subjects that execute these TPs are then capable of interpreting or executing (via an *exec* system call [1]) any program that would otherwise be accessible according to MAC and DAC. Data object accesses are similarly not restricted by ECS. These subjects can also spawn other subjects having the same ECS attributes by means of the *fork* [1] system call. Thus, this ECS configuration imposes no access control constraints beyond those associated with MAC and DAC.

Similarly, MAC and DAC checks can be set up to always succeed. This allows ECS's integrity and role-based controls to be the sole effective source of constraints. This is accomplished by labeling all objects at system low and setting their discretionary controls for world access in all modes. In addition, all roles must be defined so that they are accessible only during system low sessions.

Hybrid configurations are also possible. In particular, it may be useful to operate a single Unix-like domain in which ECS controls are not used, together with a set of tightly constrained domains tied to specific roles. This allows a single system to be used simultaneously for two purposes: 1) general purpose computing, including software development, and 2) a turn-key environment of canned programs. The former allows manipulation of uncontrolled programs and data (e.g., Clark-Wilson model Unconstrained Data Items). The latter imposes highly structured restrictions on accesses to controlled operational data. This hybrid configuration also allows software of unknown origin to be present on a system while preventing viruses in such software from infecting programs and data in protected domains.

5 ECS Architecture

The pertinent aspects of the architecture of ECS are described in the sections that follow.

5.1 Hardware Privilege States

ECS was designed to exploit existing trust features and assurances of Trusted XENIX while minimizing impact on it. This is accomplished by extending Trusted XENIX so that its architecture exports an additional hardware–defined privilege state. As shown in Figure 1, Trusted XENIX is designed to use two of the four hardware privilege states of the Intel 80286[7][10]. The Trusted XENIX kernel runs at privilege level 0, the most privileged 80286 state, while user applications and utility programs run at privilege level 3, the least privileged state. Privilege levels 1 and 2 are unused and are ordinarily hidden by the kernel interface.

The ECS design takes advantage of one of these unused privilege levels. The central component of ECS, the Access Mediation Module (AMM), runs at privilege level 1. As shown in Figure 1, the Trusted XENIX kernel has been extended to support the loading, mapping, swapping, and execution of privilege level 1 memory segments. Once ECS has been booted, these segments contain AMM code and data. The extended kernel allows the protection ring facilities of the 80286 to be employed in the manner of Multics [22] and GEMSOS [27].

In an ordinary Trusted XENIX system, access to kernel services is provided via a special call gate described by a descriptor in the global descriptor table [9]. Descriptors and descriptor tables are hardware-recognized data structures manipulated by the kernel. Kernel service calls transfer control through the gate, causing the processor privilege level to change to the level specified in the descriptor, in this case privilege level 0. As

[6] ECS does not distinguish between read and execute access.

[7] 80286 is a registered trademark of the Intel Corporation.

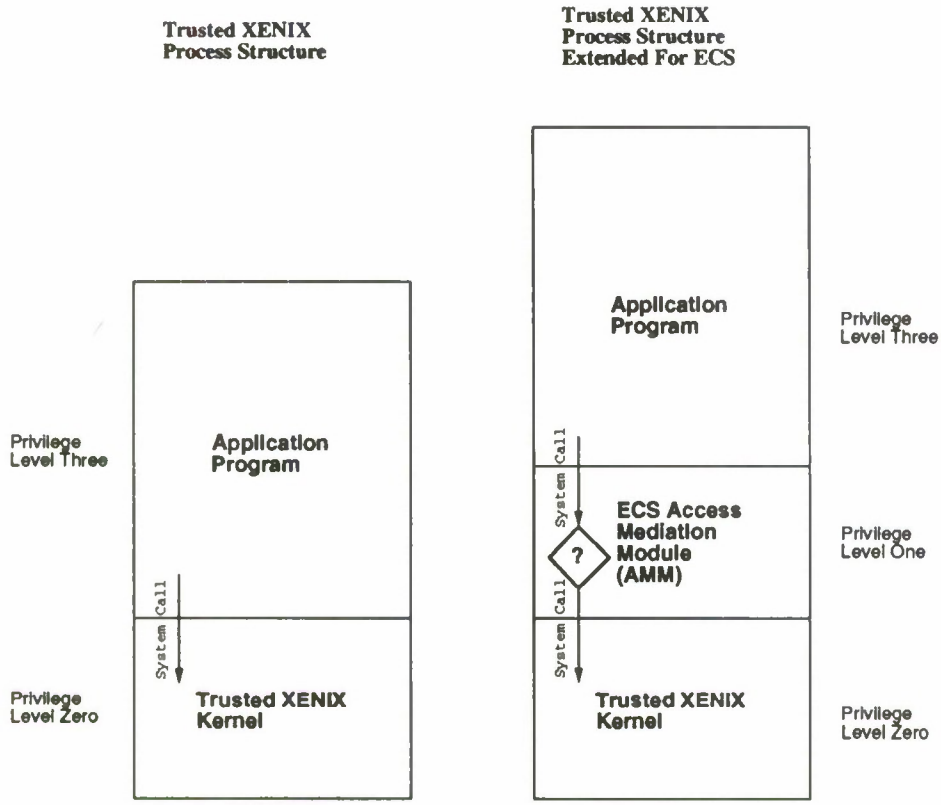


Figure 1: Trusted XENIX Process Structure

part of ECS extensions to the kernel, the descriptor table has been modified so that kernel service calls are no longer transferred directly to the kernel. Instead, they are intercepted by the AMM executing at privilege level 1. The AMM forwards to the kernel only those calls that satisfy ECS access control constraints. The AMM forwards these calls using a separate kernel call gate that is not accessible to applications programs executing at privilege level 3. Other ECS changes to the Trusted XENIX descriptor tables allow the segments containing AMM code and static initialization data to be shared automatically by every process. This saves memory by eliminating unnecessary duplication of code and static data.

The AMM component of ECS is not a separate process. Rather, each process contains its own AMM.[8] Executing the AMM at privilege level 1 gives the AMM several kernel-like capabilities. For example, the AMM within each process can access all privilege level 3 memory segments that are in that process's address space. However, unlike the kernel, the AMM in one process has no special ability to access the address spaces of other processes. A second benefit of privilege level 1 execution is that the AMM's code and data structures are protected from tampering by code executing at privilege level 3.

Prototyping has demonstrated that the extensions described above represent a relatively small change in

[8] As described later, this generalization does not apply to Trusted XENIX *Trusted Processes*.

the Trusted XENIX kernel [24]. Nevertheless, because of their presence, a configuration that includes ECS does not constitute an NCSC-evaluated configuration of Trusted XENIX. This may be an important fact in particular operational settings. However, from the research perspective of designing ECS so that it *could* be added, removed, or modified without affecting the trustedness of a suitably designed trusted system, it is irrelevant.

5.2 AMM Overview

The AMM within a process intercepts and mediates all kernel calls made by code executed at privilege level 3 (e.g., calls made by a Clark-Wilson TP). For calls that open, read, or write files, devices, shared memory, semaphores, and other Trusted XENIX objects, the AMM performs a domain check. If the requested access mode and object type are not listed in the bound operation definition associated with the process, then the AMM rejects the call. Otherwise the call is forwarded to the kernel, unless the object is a state-controlled object. If the object is a state-controlled object, additional checks are required.

Since accesses to state-controlled objects are permitted only via specialized AMM services, the AMM will reject attempts to access these objects via kernel services. The AMM services compare the process's TP name and the object's current state indicator with the state transition table associated with the object's type. If these match, the access attempt is translated into a set of kernel calls that obtain write access to the requested object, and provide mutual exclusion and current-state indicator updating. If dynamic separation of duty constraints are associated with the object's type, yet another check is performed. This check compares the separation of duty constraints with the process's user identifier and the object's mini-audit log. If the access attempt does not violate the constraint, a similar translation into kernel calls is made, including additional calls that update the mini-audit log.

The AMM controls and protects the process's subject attributes. These include its bound operation identifier and definition, which describe the subject's domain definition and allowable domain transitions. If the *invoke bound operation* service is called, the AMM determines whether the requested bound operation is invocable by the subject. If so, the AMM terminates the subject and creates a new subject in the caller's process. The AMM implements this via a call to the Trusted XENIX *exec* service [1] preceded by AMM preprocessing. The preprocessing steps include replacing the process's bound operation identifier and definition with that of the called bound operation, closing all open descriptors to objects that should not be accessible to the called bound operation, and masking out the collection of UNIX environment variables [1] that might otherwise be inherited by the new bound operation. The AMM then makes an *exec* call to the kernel. This causes the TP (program) associated with the requested bound operation to overlay the caller's TP and begin execution.

Ordinary *exec* calls that originate within a TP are handled differently. The AMM will forward these calls to the kernel as long as the ECS type associated with the requested program object is readable in the caller's domain. The AMM does not perform the preprocessing steps described above because in this case the process must continue under the auspices of the calling bound operation. The AMM also forwards *fork* service calls to the kernel. Fork calls cause the creation of new processes and hence, new ECS subjects. For these calls, the extended kernel automatically copies the calling process's entire address space into a new process. The address space includes the ECS subject attributes and other information maintained by the AMM in the process's privilege level 1 memory.

5.3 Session Start-up

At the start of each session, the start-up bound operation associated with a requested role is automatically run. This is accomplished by having the Trusted XENIX login process automatically transfer control to the ECS Login Program, a predefined bound operation that is part of the ECS TCB. The ECS Login program checks the user's authority to assume a requested role. If the user is authorized, it uses the *invoke bound*

operation service to run the start-up bound operation associated with the role.

5.4 Storage of ECS Object and Subject Attributes

The object type labels ECS uses for domain enforcement are similar in some respects to MAC labels managed by Trusted XENIX. For example, ordinary users do not have the capability to change these labels. Trusted XENIX embeds MAC labels in file system internal data structures called inodes [1, 9]; there is one inode for each file, and the inode contains that file's label. One design option for ECS was to expand the inode structure so that it could accommodate the ECS label as well as the MAC label. The inode structure, however, is so fundamental to file system operations that this option was ruled out as too invasive and harboring unacceptable risks to the reliability of Trusted Xenix.

Instead, ECS object labels, and other metadata, are kept in a collection of ordinary Trusted XENIX files. These metadata files themselves bear ECS labels that are used to protect them from tampering by ECS subjects. Since the ECS AMM intercepts and mediates all kernel calls, it can prevent these metadata files from being modified by unauthorized ECS subjects. Except for files containing role authorization lists, the only subjects authorized to modify metadata files are subjects accessible exclusively to the ECS trusted systems programmer. The AMM, however, has relatively free access to these files, and is constrained only by Trusted XENIX MAC and DAC. As described in a later section, the performance penalty associated with this design will be reduced if necessary by caching metadata in sharable privilege level 1 memory.

ECS subject attributes are maintained by the AMM in per-process privilege level 1 memory and are accessible only to the AMM that is part of that process (and to the kernel). This approach is analogous to the design of the kernel, which keeps Trusted XENIX subject attributes in per-process kernel memory.

6 Designing ECS as a TCB Subset

ECS has been designed as a hierarchical TCB subset. This section describes how TCB subset concerns influenced its design and its relationship to the Trusted XENIX TCB.

6.1 Relevance

Although TCB subsetting has been advanced primarily in the context of TCSEC, a broader applicability of the approach has been implied. For example, [30] states that "Arbitrary access control policies are admitted in the discussion ... so that arguments can be made which are valid for any access control policy." Similarly, the TDI [34] describes one of its purposes as providing a standard for "systems that satisfy trust requirements (with particular emphasis on preventing the disclosure of data) for sensitive applications."

The assertion that TCB subsetting is applicable to systems trusted to support policies other than DoD confidentiality is an important one, especially in view of the increasing interest in broader notions of trust as exemplified by the ITSEC [11], the Federal Criteria [19], and other related research and standards activities [25, 12]. As yet, this assertion is unexplored; the author knows of no published reports on the use of TCB subsetting in the design of such systems. Superficially, it may appear that ECS simply enforces an exotic form of DAC, and is no different in its use of subsetting than trusted database management systems under development [17]. ECS controls, however, are *not discretionary* in nature, providing no means for subjects to pass permission to other subjects, as suggested in the TCSEC glossary. Consequently, ECS may be the first demonstration of the applicability of TCB subsetting to policies other than DoD confidentiality.

6.2 Non-Kernel Design

Since the heart of the Trusted XENIX TCB is the kernel, an obvious design option was to implement ECS as a set of kernel extensions. If ECS were implemented in this manner, however, it would share the same execution domain as the kernel, and would need to be trusted to not subvert Trusted XENIX's MAC and DAC enforcement. This would directly contradict the objectives of the TCB subset approach.

The decision not to embed ECS object labels in the existing inode structure involved the same considerations. Inodes are metadata whose contents are controlled by the kernel. It would be difficult to provide ECS the degree of inode access it needs without giving ECS the ability to subvert the Trusted XENIX kernel. For similar reasons, ECS subject attributes are stored separately from subject attributes maintained by the kernel.

6.3 Privilege Level One Address Mappping

To support the AMM, process address spaces include four kinds of memory segments that are accessible only when the processor is executing at privilege level 1 or 0. These segments contain the following kinds of information:

- Per-Process Metadata
- AMM Code
- Static Global Metadata
- Dynamic Global Metadata

The per-process metadata segment contains the privilege level 1 call stack and the ECS subject attributes associated with the process. This segment also holds the AMM's program variables including those used to keep track of its execution state. As the name implies, a separate per-process metadata segment is allocated and mapped for exclusive access by each process; there is no sharing. This exclusivity constraint and other segment access constraints are enforced by the 80286 memory management hardware according to the segment descriptor tables constructed by the kernel.

To use physical memory efficiently and to improve throughput, the remaining three types of privilege level 1 segments are shared to varying degrees among processes. However, because ECS is designed as an untrusted TCB subset, the degree of allowable sharing must be limited according to DAC and MAC.

The AMM code segment provides the functionality to filter and forward all calls by applications to the kernel; consequently, this segment needs to be mapped into every process. To be consistent with DAC and MAC, this is permissible only if the segment is a world-readable, read-only segment, classified system-low. This poses no problem because the information contained in the AMM code segment is consistent with these attributes.

The static global metadata segment contains parts or all of the ECS access control configuration tables. These tables contain definitions of the bound operations, roles, and object-state-controls to be enforced by ECS. These tables can only be changed by a trusted systems programmer while Trusted XENIX is in single-user maintenance mode. Thus, from the standpoint of an operational configuration, this metadata is static. Like the AMM code segment, it is world-readable, read-only, and system-low, and consequently can be mapped legitimately into the address space of each process without violating DAC and MAC. Of course, it is the obligation of the trusted systems programmer who defines this metadata to assure that the sensitivity of role names, program names, and other metadata are in fact system-low. Under some circumstances it might be necessary to use fictitious aliases for such information, to serve as an unclassified "cover story".

Dynamic global metadata segments contain metadata that changes to reflect system activity. Examples include object state indicators, mini-audit logs, and ECS object labels; all of these are created and deleted in conjunction with the creation and deletion of objects. These forms of metadata are stored primarily on disk but will be cached in shared memory segments if necessary to increase throughput during mediation. Dynamic metadata segments must be writable by subjects at varying MAC levels. Consequently, they cannot be globally mapped into the address spaces of all processes. Instead, they must be segregated by the MAC levels and category sets of the objects they describe.

The MAC constraints that must be applied to the sharing of privilege level 1 dynamic metadata segments are exactly the same as those already implemented by Trusted XENIX for privilege level 3 shared memory segments [9]. This capability will be implemented if needed by extending existing shared memory services so that they also work for privilege level 1 memory segments.

6.4 Trusted XENIX Trusted Processes

The Trusted XENIX TCB includes a collection of Trusted Processes as well as the kernel. To constitute a truly untrusted TCB subset, ECS must be untrusted with respect to these Trusted Processes as well. This dictates that the AMM not be capable of intercepting calls from Trusted Processes to the kernel. If the AMM were able to intercept these calls, it could subvert their intent. Moreover, it could interrupt the trusted path between the user and the Trusted XENIX TCB.

ECS addresses this problem by providing Trusted Processes direct access to a separate kernel call gate. The kernel has been modified so that it rejects calls that enter through this gate unless the calling process holds Trusted XENIX security privileges; all such processes are Trusted Processes. A different kernel call gate is used by the AMM. Through use of hardware protection features, the gate for AMM use is made inaccessible to code segments except those running at privilege levels 1 or 0. Hence, the kernel need not check whether the caller is authorized to use this gate. This design, depicted in Figure 2, ensures that ECS AMM cannot be bypassed by its subjects. Yet it allows Trusted Processes to invoke kernel services without interception by the AMM, and without having to trust the AMM.

The TCSEC stipulates that the TCB "maintain a domain for its own execution". A strict interpretation of this phrase may require that the AMM not even be mapped into the address spaces of Trusted Processes. ECS complies with this interpretation by allowing processes to use *either* of the process address maps depicted in Figure 1. The original two-state Trusted XENIX map includes addressability of segments at privilege levels 0 and 1. This map is used by Trusted Processes. The three-state map is intended for all other processes. It includes addressability of segments at privilege levels 0, 1, and 3.

The selection of the appropriate map is programmer controlled by means of parameters supplied to the linker and loader. If an application process is erroneously linked so that it uses the original two-state map instead of the three-state map, it will be unable to call the kernel via either the Trusted Process gate or by calling the AMM, and consequently will be non-functional. The overall topology of the Trusted XENIX and ECS TCBs is depicted in Figure 3.

6.5 A View of Subjects and Objects

Analyzing a TCB architecture for conformance with TCB subset requirements requires an identification of the subjects and objects associated with each subset [34]. In this section we briefly review the relationship between the subjects and objects that are associated with the extended Trusted XENIX and ECS TCB subsets.

A Trusted XENIX subject is defined as a process, and includes an address space [9]. ECS extensions to the kernel effect a small change in this definition. A subject's address space now may consist of two logical

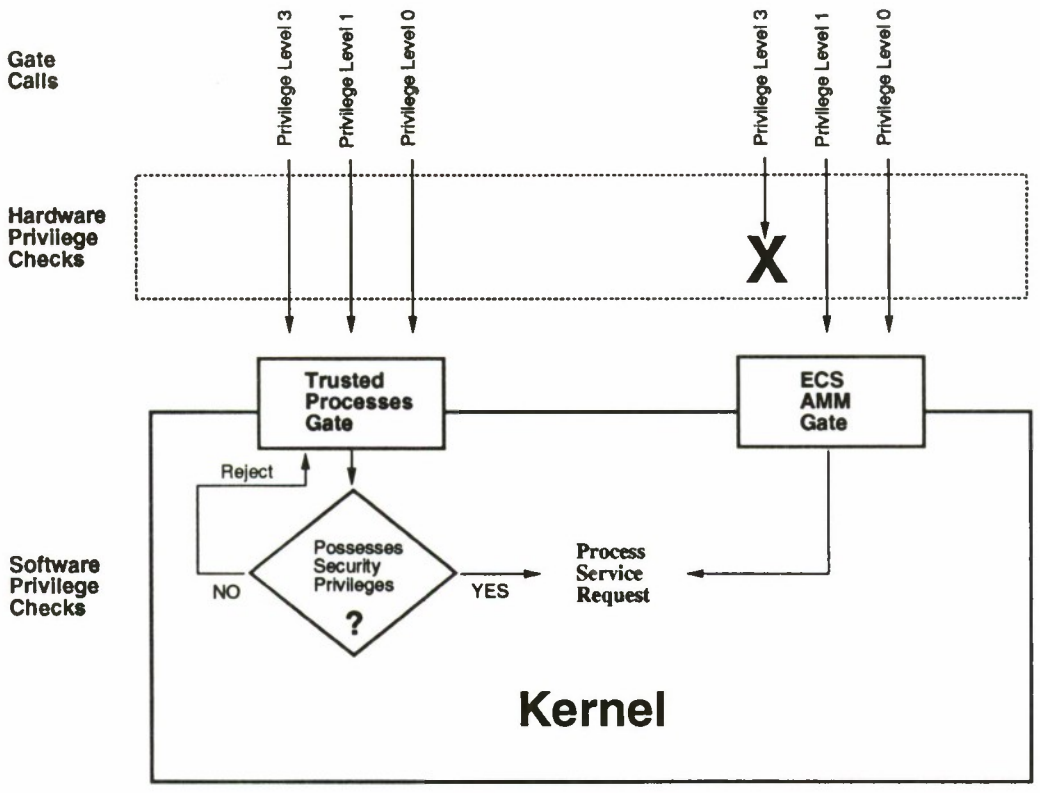


Figure 2: Kernel Gate Privilege Checks

partitions, one for privilege level 3 segments, and a new one for privilege level 1 segments. The MAC and DAC constraints governing the introduction of objects into subjects' address spaces, however, are identical for both partitions. The AMM resides in the level 1 partition of each untrusted subject. As a result, from the standpoint of MAC and DAC enforcement, the AMM is not an independent subject. Rather, it is part of each Trusted XENIX subject into which it is mapped.

An ECS subject is defined by a pair consisting of the privilege level 3 portion of a process, and an ECS bound operation definition. This definition parallels that of the TCSEC, i.e., "a process/domain pair". By this definition, the privilege level 3 portion of each process[9] is a separate subject. In addition, when an ECS subject invokes a different bound operation, the invoking subject is destroyed (or made permanently inactive), and a new ECS subject is created within the old subject's process. This view of subjects is similar to that described in [16]. The new ECS subject retains the Trusted XENIX security attributes of the old ECS subject; in fact, these two *ECS subjects*, together with the AMM, are viewed by Trusted XENIX as a single *Trusted XENIX subject.*

Trusted XENIX objects include files, directories, semaphores, shared memory, and devices. ECS re-exports

[9] Excluding Trusted XENIX Trusted Processes.

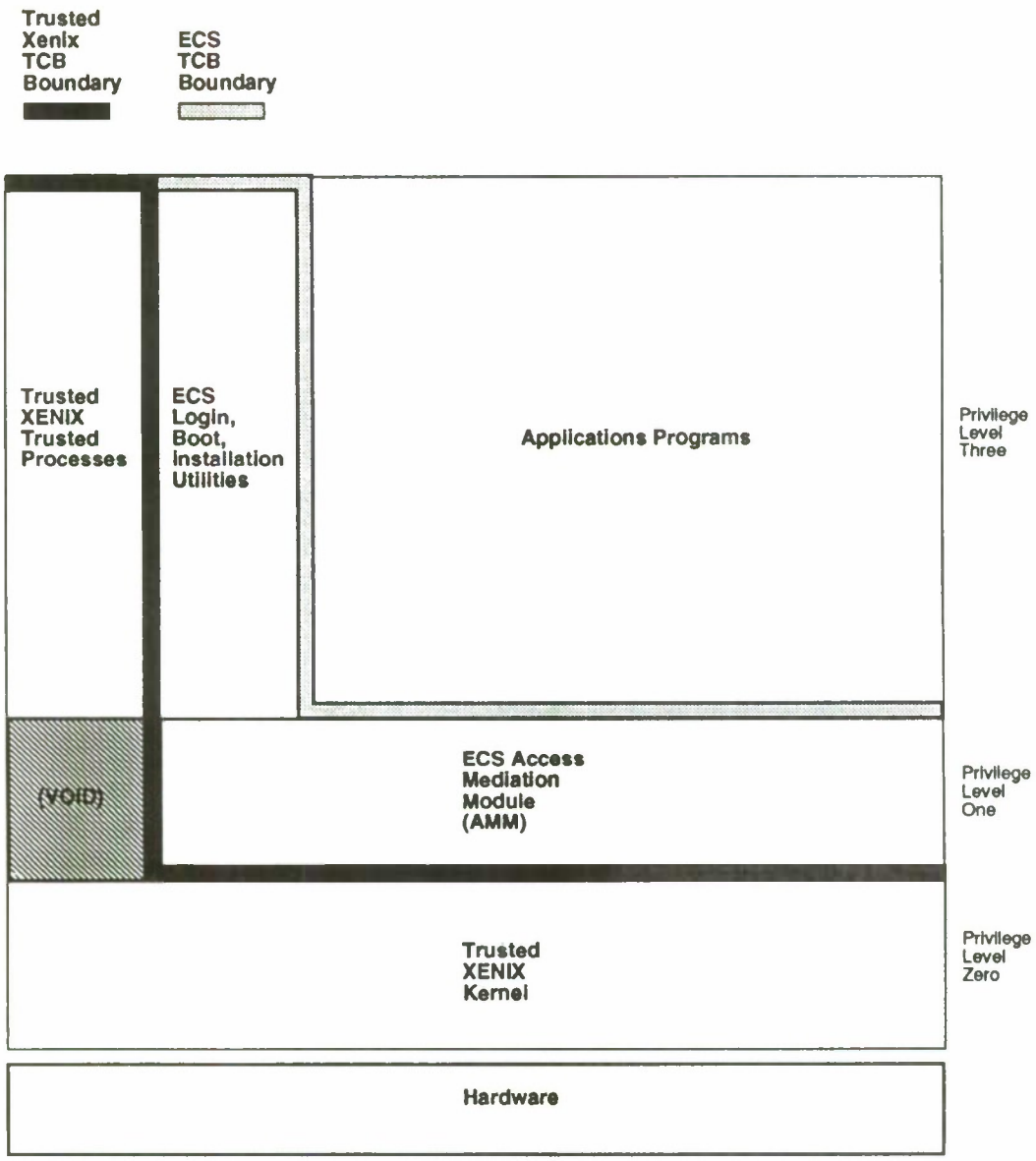


Figure 3: TCB Topology

all of these objects. In addition, it exports a new abstraction, the state-controlled object, that cannot be accessed using Trusted XENIX kernel services. State-controlled objects are an example of the "interpretively accessed" objects described in [30].

7 Summary

The Extended Access Control Subsystem (ECS) has been described. ECS is a research prototype for investigating the extension of trusted systems technology to support policies beyond DoD confidentiality. ECS provides a configurable set of access control mechanisms intended to support a range of application-specific security policies. ECS is innovative in several respects:

- It represents a synthesis of important ideas from the research literature.
- It has been designed specifically to support integrity and role-based policies, e.g., those described by the Clark-Wilson integrity model.
- It demonstrates the applicability of TCB subsetting to the design of access controls other than those for MAC or DAC.
- It has been designed for compatibility with an existing trusted system and existing untrusted applications software.

The impact of TCB subset considerations on ECS has been described; these include a non-kernel design, selective policy-governed sharing of privilege level 1 memory segments, and ECS bypass facilities for Trusted Processes. A view of the relationship between the subjects and objects associated with Trusted XENIX and ECS has been described.

The TCSEC and its interpretations, such as the TDI, have been criticized as being irrelevant for systems that support policies other than DoD confidentiality. The ECS prototype argues against this view (as do [15, 29]). It illustrates that the principles underlying the TCSEC and the technology underlying current trusted systems are *not* in contradiction with the access control requirements of other policies. Rather, these principles and technology may provide a useful base from which additional principles, technology, and evaluation criteria can be developed. This prototype also suggests that suitably designed trusted systems can be extended to support non-TCSEC policies without invalidating their TCSEC evaluation ratings. This may be a promising transition strategy for evolving TCSEC-oriented products to meet the security needs of the commercial sector.

References

[1] M.J. Bach. *The Design of the Unix Operating System*, Prentice-Hall, Englewood Cliffs, NJ, 1986.

[2] R.W. Baldwin. Naming and Grouping Privileges to Simplify Security Management in Large Databases. In *Proc. 1990 IEEE Symposium on Security and Privacy*, pages 116–132, Oakland, CA, May 1990.

[3] D. Bell and L. LaPadula. *Secure Computer System Unified Exposition and Multics Interpretation.* Technical Report MTR-2997, MITRE Corp., Bedford, MA, July 1975.

[4] D. Bell. Putting Policy Commonalities to Work. In *Proc. 14th National Computer Security Conference*, pages 456–471, Washington, DC, October 1991.

[5] W. E. Boebert and R. Y. Kain. A Practical Alternative to Hierarchical Integrity Policies. In *Proc.* 8^{th} *National Computer Security Conference*, pages 18–27, Gaithersburg, MD, September 1985.

[6] D. Clark and D. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *Proc. 1987 IEEE Symposium on Security and Privacy*, pages 184–194, Oakland, CA, April 1987.

[7] D. Clark and D. Wilson. Evolution of a Model for Computer Integrity. In *Proc.* 11^{th} *National Computer Security Conference*, Baltimore, MD, October 1988.

[8] Department of Defense. *Department of Defense Trusted Computer System Evaluation Criteria*, December 1985. DoD 5200.28-STD.

[9] *Final Evaluation Report, Trusted Information Systems, Inc. Trusted XENIX.* National Computer Security Center, Ft Meade, MD, January 1991.

[10] *286 Microprocessor Programmer's Reference Manual.* Intel Corporation, Mt Prospect, Illinois, 1990.

[11] *Information Technology Security Evaluation Criteria (ITSEC)*, Office for Official Publications of the European Communities, Brussels, Belgium, 1991.

[12] *Integrity-Oriented Control Objectives: Proposed Revisions to the Trusted Computer Systems Evaluation Criteria (TCSEC)*, C Technical Report 111-91, National Computer Security Center, Ft Meade, MD, October 1991.

[13] P. Karger. Implementing Commercial Data Integrity with Secure Capabilities. In *Proc. 1988 IEEE Symposium on Security and Privacy*, pages 130–139, Oakland, CA, April 1988.

[14] C. Landwehr, C. Heitmeyer and J. McLean. A Security Model for Military Message Systems. In *ACM Transactions on Computer Systems*, Vol 2, No.3, August 1984.

[15] T. Lee. Using Mandatory Integrity to Enforce "Commercial" Security. In *Proc. 1988 IEEE Symposium on Security and Privacy*, pages 140–146, Oakland, CA, April 1988.

[16] T. Levin, S. Padilla and C. Irvine. A Formal Model For Unix Setuid. In *Proc. 1989 IEEE Symposium on Security and Privacy*, pages 73–83, Oakland, CA, April 1989.

[17] T.F. Lunt, et al. A Near term Design for the SeaView Multilevel Database System. In *Proc. 1988 IEEE Symposium on Security and Privacy*, page 234–244, Oakland, CA, April 1988.

[18] F.L. Mayer. *Security Controls for an Automated Command and Control Information System(ACCIS): Baseline Definition.* Technical Report TISR-201, Trusted Information Systems, Glenwood, MD, May 1989.

[19] *Minimum Security Functionality Requirements For Multi-User Operating Systems (Draft)*, Computer Security Division, Computer Systems Laboratory, National Institute of Standards and Technology, Gaithersburg, MD, January 1992.

[20] W. Murray. Data Integrity in a Business Data Processing System. In *Report of the Invitational Workshop on Integrity Policy In Computer Information Systems (WIPCIS)*, Waltham, MA, October 1987.

[21] M.J. Nash and K.R. Poland. Some Conundrums Concerning Separation of Duty. In *Proc. 1990 IEEE Symposium on Security and Privacy*, pages 201–207, Oakland, CA, May 1990.

[22] E.I. Organick. *The Multics System: An Examination of Its Structure*, The MIT Press, Cambridge, MA, 1972.

[23] R. O'Brien and C. Rogers. Developing Applications on LOCK. In *Proc. 14th National Computer Security Conference*, pages 147–156, Washington, DC, October 1991.

[24] H. Orman and D. Rothmie. *Guide to TX Privilege Layer One: Implmentation and Use.* Technical Report TISR-383, Trusted Information Systems, Glenwood, MD, September 1991.

[25] J.E. Roskos, S.R. Welke, J.M. Boone, and T. Mayfield. *Integrity in the Department of Defense Computer Systems.* Technical Report P-2316, Institute for Defense Analyses, July 1990.

[26] R. Sandhu. Transaction Control Expressions for Separation of Duty. In *Fourth Annual Computer Security Applications Conference*, pages 282–286, Orlando, FL, December 1988.

[27] R.R. Schell, T.F. Tao, and M. Heckman. Designing the GEMSOS Security Kernel for Security and Performance. In *Proc. 8th National Computer Security Conference*, pages 108–119, Gaithersburg, MD, 1985.

[28] Lawrence J. Shirley and Roger R. Schell. Mechanism Sufficiency Validation By Assignment. In *Proc. 1981 Symposium on Security and Privacy*, pages 26–32, Oakland, CA, April 1981.

[29] W.R. Shockley. Implementing The Clark/Wilson Integrity Policy Using Current Technology. In *Proc. the 11th National Computer Security Conference*, Baltimore, MD, October 1988.

[30] W.R. Shockley and R.R. Schell. TCB Subsets For Incremental Evaluation. In *Proc. Third Aerospace Computer Security Conference*, pages 131–139, Orlando, FL, December 1987.

[31] D. Sterne, M. Branstad, B. Hubbard, B. Mayer, L. Badger, and D. Wolcott. *Security Policies for Tactical Environments: A Research Study.* Technical Report TISR-353, Trusted Information Systems, Glenwood, MD, November 1990.

[32] D. Sterne, M. Branstad, B. Hubbard, B. Mayer, and D. Wolcott. An Analysis of Application Specific Security Policies. In *Proc. 14th National Computer Security Conference*, pages 25–36, Washington, DC, October 1991.

[33] D.F. Sterne. On The Buzzword "Security Policy". In *Proc. 1991 IEEE Symposium on Security and Privacy*, Oakland, CA, May 1991.

[34] *The Trusted Database Management System Interpretation of of the Trusted Computer System Evaluation Criteria.* NCSC-TG-021, National Computer Security Center, Ft Meade, MD, April 1991.

[35] D.J. Thomsen. Role-based Application Design and Enforcement. In *Proc. of the Fourth IFIP Workshop on Database Security*, Halifax, England, September 1990.

[36] D.J. Thomsen and J.T. Haigh. A Comparison of Type Enforcement and Unix Setuid Implementation of Well-Formed Transactions. In *Proc. Sixth Annual Computer Security Applications Conference*, pages 304–312, Tucson, AZ, December 1990.

[37] *Trusted Mach System Architecture*, TISR-324, Trusted Information Systems, Glenwood, Maryland, October 1991.

A TOOL FOR COVERT STORAGE CHANNEL ANALYSIS OF THE UNIX KERNEL

David A. Willcox
Steve R. Bunch

Motorola Microcomputer Group
Urbana Design Center
1101 E. University Ave.
Urbana, IL 61801

Motorola MCG has developed a tool that was used to search for covert storage channels in the kernel of USL's security-enhanced UNIX System V Release 4 (SVR4.1ES). Using parse trees generated by the C compiler front-end, the tool does a control and data flow analysis of the entire kernel to detect references that violate system security policy. Hints (assertions) added to the kernel source give the tool additional information about security relationships.

This paper describes the covert channel analysis tool and its application to the SVR4.1ES kernel. It also presents some of the results of the covert channel analysis, and describes several sample covert channels. A method that could be used to exploit each channel is given, as is the method ultimately used to treat each channel.

Keywords: Covert channel, storage channel, data flow analysis.

INTRODUCTION

Motorola MCG worked with UNIX System Laboratories (USL) to develop the SVR4.1ES operating system, which is in formal evaluation at the B2 security level. One of the requirements of a B2 evaluation is a thorough search for covert storage channels.

Motorola MCG developed a Covert Channel Analysis (CCA) tool to assist in this search. This made practical a thorough search for covert channels in a complex, commercially-available system. This paper describes our approach and reports on our experience with using it.

COVERT CHANNELS

A covert channel is any method that allows a transfer of information in violation of the system's security policy. It allows an *agent* (a process or user) that has access to high-security information to send information to another agent that is not authorized to have that information; for example, it might allow a TOP SECRET agent to leak information to an agent that was only cleared to see UNCLASSIFIED data.

The presence of a covert channel does not automatically allow an agent to gain access to information that it should be denied, but it does allow a Trojan horse to leak information to an agent that shouldn't have it.[1]

Covert channels can be broadly broken down into two categories: storage channels and timing channels. A covert storage channel allows two processes to exchange information via some object to which they both have access. The shared object could be a storage location (a kernel global variable for example), or it might be some attribute of an object (such as a file) that the two processes can legitimately share but not write. In a storage channel, the receiving process can read a value set (directly or indirectly) by the sending process.

A covert timing channel occurs when the sending process can affect the time it takes the receiving process to perform some operation. A classic example is

UNIX is a trademark of UNIX System Laboratories, Inc.

Copyright 1992, Motorola, Inc. All rights reserved.

[1] If the security policy is properly designed and there are no covert channels, then a Trojan horse cannot leak information. If a "low agent" fools a "high agent" into running a Trojan horse program, the program might do damage to the high agent's files, but won't be able to leak any information to the low agent.

reading a block from a file, causing the disk arm to be repositioned and thus altering how long it takes another process to access its own file on the same disk.

Note that to be of concern, a covert channel does not have to transfer a large amount of data at once. Many covert channels can transmit only a single bit on each use. Often the information is simply that the sender did or did not perform some operation. But, if the channel can be used repeatedly, or if there are many different instances of the same channel (for example if it can be exercised independently on each of several different files), then there is a potential for transmitting a large amount of data. The National Computer Security Center (NCSC) guidelines for a B2 system prohibit any channel with a sustained bandwidth of over 100 bits/second. Lower bandwidth channels are permitted, but any over 10 bits/second must be audited by the system, and any over 1 bit/second must be documented.

Also note that a channel does not have to be completely reliable. Many channels are very "noisy" because they depend on close synchronization between the sender and receiver or on exclusive access to a widely-used resource. A busy system can make synchronization difficult or can generate competing uses of the shared resource. However, the techniques for error correcting codes in the presence of noise are widely known, so it will always be possible to transmit some amount of "clean" data even on a very noisy channel. Furthermore, error detection and retransmission can be quite easy in the case of covert channels because the backwards channel from receiver to sender is often "overt". (An UNCLASSIFIED agent is allowed to give information to a TOP SECRET agent, for example.) The bandwidth limit mentioned above applies to the number of "clean" bits transmitted, not the raw bandwidth of the channel.

EXISTING ANALYSIS METHODOLOGIES

Several existing techniques for detecting covert storage channels are based on the Shared Resource Matrix Methodology (SRMM). A variant of this method was used, for example, in the evaluation of Secure XENIX [Ts87]. SRMM searches for shared objects that can be written by one user and read by another.

To find covert channels using this method, one constructs a matrix with all system variables on one axis, and all system calls on the other. The variables might be simple data items of which only a single copy exists in the system, or they might be instances of a class of variables of which there are many copies. In each cell of the matrix, one notes whether the given variable can be read and/or written by the given system call. A closure operation is then performed on the matrix to account for information that can be transferred from one variable to another by a system call. (If system call *A* reads variable *x* and writes *y*, and system call *B* reads variable *y*, then *B* is considered to indirectly read *x*.)

Any variable that can be both read and written (by the same or different system calls) is a potential channel. The investigator must examine each such variable to determine if a covert channel exists. Most channels identified by this method are not covert channels, because either:

(1) They are "overt" channels. They are either transferring information within a process, or between processes that are allowed to communicate under the security policy, or

(2) They are prohibited by the system. Sufficient tests exist in the code to ensure that processes that should not be able to communicate cannot share this variable.

While SRMM can uncover covert channels, it requires a large amount of manual analysis. It was used in the analysis of user-level commands in SVR4.1ES, because there is a relatively small number of objects that can be referenced. But in a system as large as the kernel, there are thousands of variables and hundreds of system calls. Virtually all of the variables are both read and written *somewhere*, and therefore must be investigated as possible covert channels. While techniques exist to reduce the effort (some variables are local to a process and can be discarded, and others can be grouped into classes that are manipulated together), the analysis is still an expensive proposition. The theory does not help you to distinguish between covert and overt channels; much of the hard work is still "an exercise for the reader."

Another approach is the lattice model of secure information flow [De76] [De77]. This model looks at the operations on objects in a program and the *security classes* of those objects to identify implicit and explicit information flow between objects, and to flag flows that violate the security policy. This is very similar to the approach we took, but it does not address one of the biggest problems in analyzing a large, general-purpose operating system: determining the set of objects to which a pointer can refer, and tracking that set through the lifetime of the pointer variable. Discovering and tracking the security relationships between objects is the main function of the CCA tool.

THE CCA TOOL

In our analysis of the security enhanced SVR4 kernel, we search for object references that might violate the security policy, and then determine which of such references create covert channels. Identifying these "illegal" references is the purpose of the CCA tool.

TOOL GOALS

The design of the tool depends on one underlying assumption: every object in the kernel has some *security label*,[2] whether explicit or implicit, and therefore every object has some security relationship to the currently running process.

Going back to the basic requirement of a secure system, we assert that the process is allowed to read memory objects only if the process's security label *dominates*[3] the label of the object, and it is only allowed to write to an object if the object's label dominates the process's.[4] (If the labels are the same, then the process and the object dominate each other.)

The ultimate goal of the tool is to examine each data reference in the kernel and determine the relationship between the process executing and the referenced object. If the relationship cannot be determined, or if the relationship is not appropriate for the type of access (read or write), the tool generates a record of the reference, identifying the object referenced, the type of reference, and the context of the reference (the source line of the reference and a "stack traceback").

The list of "illegal" references from the tool has to be analyzed manually. However, part of the analysis process puts tool directives into the code, and these directives can be used in later reanalyses.

Our approach is similar to the lattice flow model, the main difference being one of emphasis. The lattice model looks for illegal information flows between objects with known security labels. The CCA tool's main function is to examine references (usually via pointers) that might refer to objects with different labels, and ensure that all of the references are permitted.

A useful point made by Denning is that when the security labels of objects are static, information flows in the absence of explicit flows can safely be ignored [De77]. Since we assume that objects have static (if unknown) labels, this gives us confidence that our analysis, which does not explicitly address indirect flows, is thorough.

TOOL ORGANIZATION

The tool can be broken down into six different areas which are described below.

The parser. The parser for the CCA tool is derived from a C compiler. It parses the kernel source and produces files containing expression trees and symbol table information for each function.

The structifier. The structifier reads in the parse information generated by the parser. The form generated by the parser is inherently unstructured; the control structures are conditional and unconditional GOTOs. It is much easier to analyze the code when the control structures are simple, so the structifier converts the parsed functions into a form that uses only IF-THEN-ELSE and WHILE-DO. The resulting code is often larger, since sections of code must be duplicated, but it is easier to analyze.

Value tracking. The value tracking code is the central part of the CCA tool. The tool can deduce relationships to objects only if it has some knowledge about the objects to which a pointer can point. The value tracking code maintains this information. For each variable or structure field, the value tracking code keeps a list of possible values. Multiple values are possible since there can be multiple control paths that lead to the expression being evaluated.

Each value is a pointer value or simple numeric scalar value. A pointer value includes the following information:

(1) The identity of the variable pointed to

(2) Any subfield or substructures explicitly addressed

(3) An indication that the "pointee" is an element of an array

(4) Any knowledge about security relationship between pointer and pointee (see DETERMINING RELATIONSHIPS).

While pointer values are most important, simple numeric values are also kept. These are primarily used to allow the tree walking code (see below) to detect cases where conditionals must evaluate to TRUE or

---

[2] A security label identifies the hierarchical security level such as UNCLASSIFIED or TOP SECRET, plus special-purpose "need-to-know" categories, that an agent is allowed to access.

[3] Informally, label *A* dominates label *B* if an agent with label *A* may see anything that agent *B* may.

[4] SVR4.1ES permits write access to an object only if it has the same label as the process. This is to avoid the covert channels caused by status returns from write operations. The CCA tool does not enforce this, but detects channels that would result from any returned status.

FALSE. There are many cases in the SVR4.1ES kernel where a subroutine takes a flag argument that greatly changes its operation, and it is important to be able to account for this. As a special case, the value UNKNOWN is kept to identify cases where a variable has a numeric value, but we don't care to maintain the specific value.

A major function of the value tracker is to maintain *dynamic control flow contexts.* This feature is used by the parse tree walker to keep track of values that are generated at different control flow levels in the code. It is necessary to identify the control flow context where a value was generated so that values can be properly saved and merged around control constructs. For example, if a particular possible value of a variable does not exist before an IF-THEN-ELSE construct but is generated within the THEN section, then the value tracking code ensures that the value does not appear while evaluating the ELSE section, and does appear when evaluating code following the IF-THEN-ELSE. The value tracker also keeps track of "evaluation level" so that variables in different evaluations of a function (especially recursive evaluations) are distinct.

Parse tree walker. Once the functions have been "structified", the parse tree walker traverses the resulting tree, keeping track of variable values at each point in the tree. The tree walker is responsible for interpreting the semantics of the parsed code, including the effects of the control constructs. It makes calls to the value tracking code to find current values for variables, to record new values, and control the dynamic control flow context. When the tree walker encounters an IF-THEN-ELSE construct, it attempts to determine if the conditional expression must evaluate to TRUE or FALSE in the current context. If the result must be one or the other, the tree walker evaluates only the THEN or ELSE of the conditional. If the value of the conditional cannot be determined, then the THEN and ELSE branches must both be evaluated, each with the same starting context, and the values resulting from the two cases must be merged to form a new context. The handling of a WHILE-DO construct is similar. Unless the conditional can be shown to be FALSE, the body of the loop is evaluated, and the effects on variables produced by the loop are merged back into the state that existed at the beginning of the loop. However, there is a difference; if there are any values generated within the loop that were not visible at the beginning of the loop, then it is possible that the new values would affect the results of the loop. The loop must be reevaluated until no new values are generated.

MAC policy. The Mandatory Access Control (MAC) policy code is responsible for maintaining knowledge about relationships among objects, and for reporting illegal references to objects. It is called by the value tracking code whenever a value is stored or loaded, and whenever pointers are manipulated.

Command interface. A very simple command interpreter is included to control operation of the tool. Using this interface, it is possible to turn various options on or off, to specify default security relationships, or to evaluate specific routines.

DETERMINING RELATIONSHIPS

The process and an object can have any one of four possible relationships:

(1) Labels equal

(2) Process "strictly dominates" the object

(3) Object "strictly dominates" the process

(4) Neither dominates the other (incompatible labels)

Similar relationships can exist between any other two objects in the system. Strictly speaking, the tool does not determine which relationships *are* possible. Rather, it assumes that any are possible, but then strives to show that some are not possible. Therefore, if we say that the tool knows that the process dominates an object, then we really mean that it has determined that cases (3) and (4) cannot occur.

The tool uses several methods to determine the relationship between the process and a referenced object:

Stack variables. Local stack variables are assumed to be private to the process, and therefore to have the same label as the process. If there is a path that allows a process to reference another process's stack variables via a pointer, then that would be subject to the same checks as any other pointer dereference as described below.

Fixed relationships. Other, non-stack variables can be specified to have a fixed relationship to the process. In the SVR4.1ES kernel, for example, the *u* structure is private to the process, and therefore is always considered to have the same label as the process. Other variables can be specified to dominate or be dominated by the process when those variables are expected only to be written or read by processes.

Such relationships are specified in a control file read by the tool before performing the analysis.

Assertions in evaluated code. Other relationships can be assumed to be true only after appropriate checks have been made in the code. For example, when opening a file, the relationship between the process and the file is not known until an explicit check is made of the labels associated with the file and process. Before such a check is made, any reference to data structures associated with the file are potential covert channels that must be reported and investigated. After the check has been made, only references contrary to the relationship indicated by the check need be reported.

Theoretically, the tool could recognize the security checks that appear in the code, and adjust its knowledge of the relationship appropriately. This was, however, impractical. Instead, MAC_ASSERT macros were inserted at appropriate points into the code being analyzed to tell the tool about known relationships. These macros are no-ops when the kernel is compiled for execution, but expand into special tool directives when the kernel is parsed for analysis. (Approximately 100 such macros were needed in the SVR4.1ES kernel.)

Deduced relationships. When a pointer value is stored via another pointer (as when a pointer to one data structure is stored into another), the tool deduces the relationship between the two data structures based on the relationships between the process and each of the structures. For example, if p and q are local pointers to data structures (meaning that the pointers themselves are at the same level as the process), and the tool encounters the assignment $p\rightarrow x=q$, then it deduces the relationship between $p\rightarrow x$ and $*(p\rightarrow x)$. If, for example, the process dominates $*q$ and is at the same level as $*p$, then $p\rightarrow x$ dominates $*(p\rightarrow x)$.[5]

Similarly, the tool deduces relationships when a pointer value is loaded via another pointer. Using the above example, if the tool next encountered a statement $r=p\rightarrow x$, it would know that the process dominates $*r$.

Pointer arithmetic. Arithmetic on a pointer can destroy what is known about the relationship between the pointer and the object it addresses. Suppose that the tool knows that a particular pointer points to an element of an array, and the relationship to the particular element is known. If the pointer is incremented or decremented it still points to an element of the array, but the security relationship is not necessarily the same as the original element. (We assume that the code being evaluated is well enough behaved to not address elements outside of the array.) The CCA tool resets the known security relationship back to the fixed default for the array.

[5] The notation $p\rightarrow x$ is from the C language. It means "item x in the data structure whose address is in p." The notation $*p$ means "the data structure whose address is in p."

Post-Run Analysis

When a run of the CCA tool is completed, the results must be analyzed. Each read or write that the tool cannot determine is safe is a potential covert channel. Note that it is not necessary that the tool report both reads and writes to an object for that object to be part of a covert channel. If, for example, the tool finds many reads that were permitted by the policy, and a single write that was not, then only the write will be reported. Since the write might store sensitive information that should not be available to the lower-level process (if nothing else, the sending process can communicate information by controlling whether the write occurs at all), each of the reads could potentially be providing information to the receiving process that it is not authorized to have.

Each illegal access reported by the tool could be due to any of the following:

(1) It might truly be a covert channel. In this case, the channel must be reported to the evaluation team.

(2) It might be benign. There are cases where variables are accessed at different levels, but it can be shown that no communication is possible. Other variables are only read (for example, system configuration parameters) or written (for example, metering information only readable by a privileged agent) during normal operation. It is appropriate to add MAC_ASSERTs or default relationships for such variables. This reduces the manual analysis required by eliminating the reports generated in subsequent runs of the tool.

(3) It might be that the tool wasn't smart enough to figure out that a particular access was legal. In many cases, this can be remedied by adding new MAC_ASSERTs to the kernel. There are cases where there just is no appropriate place to add a MAC_ASSERT (the SVR4.1ES kernel code is very complex), and these cases must be recorded as not a problem. No report to the evaluation team is necessary.

When new MAC_ASSERTs or default relationships were added to the kernel, the tool was rerun, producing a new report. The new report would be smaller than the first, but it still had to be examined to be sure that the remaining reports are not due to covert channels. The procedure was repeated until all accesses reported by the tool can be categorized as one of the above three

cases.

Since we completed our work, Porras has described *covert flow trees* as a technique for covert channel analysis [Po91]. While we believe that our approach has an advantage in that it immediately eliminates references to objects that are known to not cause covert channels, it is possible that covert flow trees would be a valuable tool to aid the post-run analysis. It could be used to identify explicit reference patterns that could be used to exploit a covert channel.

EXPERIENCE

Using the CCA tool is very much an "expert mode" operation. Picking appropriate places to insert MAC_ASSERTs and getting the tool initialized requires knowledge of how the tool works.

COVERAGE

Running the tool on the entire unmodified kernel proved to be impractical. Some of the difficulties we ran into are:

(1) It is impractical to let the kernel initialize itself. This is due partly to the fact that many tables are initialized by system initialization in a way that cannot be understood by the tool, and due partly to the amount of code in the complete kernel.

(2) Each subroutine is reevaluated in each context in which it is called, and many subroutines are called from many places in a given caller. This can produce an exponential growth in the number of times a routine is evaluated as the calling stack grows. We've observed at least 32 levels of subroutine calls, and routines near the bottom of the calling tree are evaluated hundreds or thousands of times for each evaluation of the routines near the top of the tree.

(3) Some very complex routines compound the above problem by evaluating subroutines *many* different times.

(4) There are some expensive, but not very interesting (from a security policy perspective), routines that are called from many places.

(5) There are hardware dependencies in some routines.

We used a number of techniques to circumvent these problems in our analysis.

Initialization routines. Rather than have the tool run through all of the kernel initialization routines, we used special-purpose initialization routines. This initialization ignores most of the complexities of initializing the true kernel. Instead, it created dummy data structures that substitute for the ones actually created by kernel initialization. References to these dummy structures were analyzed as if they were references to the real structures.

Simplified routines. We replaced some of the more complex routines with simplified versions during the analysis. These simplified versions performed most of the same work and called the same subroutines as the originals, but the looping structure was simplified to reduce the number of calls to subroutines.

Stub routines. Some expensive routines that are not interesting for security were disabled. The *panic*() routine, which abnormally terminates kernel execution, is an example.

Single evaluation. There are some very expensive routines that are called in many places, but that cannot be ignored in the analysis. These routines were evaluated a few times in different contexts to find possible violations generated by those routines or their callees. Then, for the overall kernel analysis we replaced those routines with stubs that have equivalent effect (from a security standpoint), but that are much faster to evaluate.

COVERT CHANNELS IN SVR4.1ES

The covert storage channels found in the analysis of SVR4.1ES fall broadly into four categories:

(1) Shared identifiers.

(2) Resource exhaustion.

(3) Caches.

(4) Direct covert channels.

Approximately 65 channels were found overall: 10 shared identifiers, 12 resource exhaustion, 8 caches, and 35 direct. Of these, 24 were eliminated through various means and 7 required no treatment because of low bandwidth. The remaining 34 required treatment to reduce their bandwidth and/or to audit their use. The following sections describe some of the channels and how they were treated.

SHARED ID CHANNELS

Shared ID channels occur when the identifier space for some resource is shared among all security labels of the system. If the allocation scheme of these identifiers is predictable, and the identifiers are available to a user, then they provide a covert channel. In general, a

sending agent can consume or release identifiers in the identifier space by allocating or freeing the appropriate resource, and the receiving agent can detect which (or how many) identifiers were used by the sender. Here are some examples:

Process ids. In the SVR4.1ES system, each new process is assigned a unique process ID (pid). The pid is the next higher unused one after the previous one allocated. A sending process can transmit $\log_2 N$ bits of information by creating between zero and N-1 new processes. (Each new process immediately terminates.) The receiving process can then create its own new process, note the pid of the new process, and determine how many processes the sender created.

The bandwidth of this channel was estimated at 25 bits/second.[6] Its bandwidth was reduced to a negligible level by randomizing the order of pid assignment.

IPC objects. The identifiers for semaphores, shared memory segments, and message queues in SVR4.1ES are all indexes into fairly small tables within the kernel. These identifiers are common to all security labels, and they are allocated on a "first free" basis. To exploit this channel, the sending agent first allocates N of one of these objects. It can then transmit N bits by releasing those objects with identifiers that correspond to '0' bits, and holding the rest. The receiver then allocates objects, determines the identifiers it gets, and converts each into the number of one of the '0' bits. The receiver then frees the objects it allocated and the sender reacquires them in preparation for transmitting another N bits. This channel was estimated to transmit information at well over 100 bits/second.

The channel was reduced by changing the allocation scheme to give preference to an identifier that was previously used by a process with the same security label as the caller. If an identifier from another label must be used, the system audits that fact, and ensures (by keeping a cumulative count of such occurrences) that the bandwidth ceiling isn't exceeded.

Resource Exhaustion Channels

Resource exhaustion channels are similar to shared identifier channels in that they involve a resource pool that is common to all security labels. However, in this case there is no user-visible identifier for a particular instance of the resource. In general, the sender using such a channel consumes all of the resource, and then releases a specific amount of it. The receiver then consumes the remaining resource to determine how much the sender released.

Many shared id channels can also be exploited as resource exhaustion channels, though usually at a lower bandwidth. Here are some other examples:

File record locks. SVR4.1ES supports file record locking to allow cooperating processes to coordinate access to a file. Since obtaining a record lock requires write permission on a file, this mechanism does not, in itself, produce a security hole. However, there is a fixed, maximum number of record locks that can be in existence system-wide at one time, and a single process can easily consume them.

To exploit this channel (and most resource exhaustion channels), the sender first uses up all of the system's capacity for record locks. It then can transmit $\log_2 N$ bits by releasing between 0 and N-1 of the locks. The receiver then counts how many locks it can create, and interprets the result as an $\log_2 N$-bit number. It was estimated that this could transmit as much as 180 bits per second.

The bandwidth of this channel was reduced to an acceptable level by auditing the exhaustion of file locks, and by keeping a running count of such exhaustions and delaying the process if they occur too often.

CPU exhaustion. CPU cycles are a resource that can be exhausted, and this is probably the most insidious channel that we identified. It produces a fairly high-bandwidth channel that is almost impossible to eliminate. It is possible to transmit N bits of information on each clock tick, where N depends on the particular hardware. Both the sending and receiving processes use the SVR4.1ES *setitimer*() facility to request that they receive a signal on each (100 HZ) clock tick. When the sending process receives its signal, it transmits a number n from 0 to N-1 by iterating a simple loop $S_1 n$ times. It then uses the *pause*() function to suspend itself until the next signal. (Note that because the sender is never running at the end of a clock interval, it never gets charged for any CPU usage. It therefore is given higher priority by the SVR4.1ES scheduler, and will always preempt the receiver at the end of each clock interval when it is awakened by the next signal.)

---

[6] All bandwidth numbers are for the specific USL system that is being evaluated by NCSC. The system is assumed to be idle. Many of the channels will have higher bandwidths on faster processors, and lower bandwidths on busy systems.

The receiving process counts how many times it can iterate its own simple loop before receiving the signal marking the end of the clock interval. If it went through the loop m times, it can compute n from R_1-R_2m. Optimal values for S_1, R_1 and R_2 can be determined experimentally. We were able to transmit 500 bits/second on an idle system, with about 10% of the bits received in error. With error detection and retransmission, and given that most of the erroneous bits were received in blocks, it should be possible to send 300-400 bits/second cleanly on a relatively slow processor.

We weren't able to identify any satisfactory treatment for this channel that would not degrade system performance or change the system functionality offered to applications. This channel is extremely sensitive to other system activity, and is only practical to exploit on a system with very light usage. However, many systems are very lightly used during some hours of the day.

CACHE CHANNELS

Many of the covert channels we found were produced by software-constructed caches. These are not hardware caches, but rather are in-memory copies of various items from disk. This demonstrates one of the problems of designing a secure system: many features that are added to increase system performance also make it easier to leak secure information!

In general, such caches form a class of resource exhaustion channel. If the sending process can bring into the cache an entry that can be seen by other processes (or can cause enough cache misses that other processes' entries are discarded), and if the receiving process can detect that there was a cache miss, then there is a potential for a covert channel. In some cases, a miss can be detected by noticing that the process suspended during the reference to the object, allowing another process to run. In other cases it requires measuring the time it takes to do the reference.

Directory name lookup cache. The system keeps a cache of names that have recently been found in directories. Whenever a directory is searched, the system first checks the directory name lookup cache (DNLC). In general, directories can be read from many different security labels. A sending process could therefore bring an entry into the cache by doing a lookup on that directory entry. The receiving process could then look up that same entry and time whether its lookup required a disk access.

This channel was estimated at about 66 bits/second. It was partially reduced by changing the method of selecting which entry to reuse when a new entry is to be brought into the cache. The selection now gives preference to an entry that was first looked up by a process with the same security label as the one doing the current lookup. If a cache entry previously created with a different label must be used, that fact is noted. If this occurs too frequently, that fact is audited.

Page cache. SVR4ES treats main memory primarily as a cache of information on disk. Pages are read in from disk when needed, and freed from main memory (perhaps after the data is written back to disk) when the memory is needed to hold a copy of some other disk block. Since files can be shared (read-only) by processes with different security labels, and it is possible to detect if a reference to a page required a disk access, this produces a covert channel.

To use this channel, two processes both open a shared file. To send a '1', the sending process references a page of the file, bringing it into memory. To send a '0', it does nothing. The receiving process then references the same page, and notes whether the reference occurred immediately or required a disk access. This sequence can be repeated for each block of the file, or with other files.

Because each bit transferred requires a disk access, this channel is limited by the rate at which blocks can be read from disk. With typical disk controllers and drivers, this won't be more than about a page per revolution, yielding a bandwidth of about 60 bits/second per disk. Given a more intelligent controller that buffers entire tracks and returns individual blocks to the driver on demand, the bandwidth could be much higher.

This channel would be very difficult to eliminate without completely invalidating SVR4ES's virtual memory scheme. Fortunately, the bandwidth is low enough that it is only necessary to detect possible uses of the channel; it doesn't have to be eliminated. It was treated by remembering the label of the process that first brought any given page in from disk, and then auditing the first use of that page by any process with a different label.

DIRECT COVERT CHANNELS

A number of covert channels were discovered whereby processes could directly manipulate some object that could be viewed by other lower level processes. Some of these were well known system features that had not been covered by the design of the security enhancements. Some were "undocumented features."

accessed" time of a file is updated each time the file is read, and any process that can read a file can obtain its last accessed time. Since processes with different security labels can read shared files, this produces a covert channel. Given *N* shared files, the sending process could send *N* bits of information by reading those files that correspond to '1's and not reading those corresponding to '0's. The receiving process could then look at the last accessed times of all files to determine which had been accessed.

Unfortunately, this channel is fairly deeply buried in the definition of SVR4[7]. A system that did not update the access time as described would not strictly follow the various specifications of the SVR4 interfaces such as POSIX.[8] It is also very high bandwidth; we estimated it at over 1000 bits/second. We considered several proposals to reduce the bandwidth to an acceptable level, but all were either easily subvertable, or would have a disastrous effect on overall performance.

In the end, it was decided to depart slightly from the standard behavior. When the security features of SVR4.1ES are enabled, the last accessed time of a file is not set when it is read by a process with a different security label from the file. Since relatively few programs actually make use of the last accessed time, it was felt that this was not too large a penalty to pay to eliminate the channel.

Object reuse problem with semaphores. The *semop*() and *semctl*() system calls manipulate semaphores used for interprocess communication. Both of these calls made use of a static kernel buffer as a staging area for a block of data that was copied to or from the calling process. On a "set" operation, information was copied into the static buffer from the user process and then used within the kernel. On a "get" operation, the static buffer was filled in with the requested data, and then it copied into the user's buffer.

Under normal conditions, this would not be a problem. Unfortunately, the structure defining the data to be transferred had several slack areas to preserve alignment. These slack areas were never explicitly cleared by the kernel, but were copied to or from the user's buffer along with the rest of the structure. On a "get" operation, the slack areas copied to the user would contain the values from the last "set" operation done by any user. This creates a fairly obvious covert channel.

This channel was easy to close. The system was changed so that the temporary buffer is on the kernel's per-process stack instead of being a static shared among all processes. This particular example pushes the definition of covert channel. Some would call it a security hole. We mention it to show why it is necessary to do the covert channel analysis on the final implementation of the system, not just on a high-level design.

CONCLUSIONS

The Covert Channel Analysis Tool proved very helpful in identifying objects in the SVR4.1ES kernel that could potentially be exploited as covert channels. However, as with any existing covert channel analysis method, finding such a shared object was only the first part of the analysis. Once a shared object was identified, it often took a fair amount of ingenuity to find an optimal scheme for exploiting the channel, and then to estimate the resulting bandwidth. (To have actually written a program to optimally exploit each would have been prohibitively time consuming.)

After initial development and testing of the tool, the analysis of the first internal release required approximately 10 staff months. This was a preliminary analysis of an early internal release, and much of this time was spent debugging and enhancing the tool and gaining experience with its use. Subsequent reanalyses went faster, even though they were more thorough and required documentation of the results.

Our analysis of the kernel uncovered approximately 65 covert storage channels. Of these, about 24 were completely eliminated, and another 7 required no treatment (beyond documentation) because the bandwidth was low.

Even systems that have already undergone some significant effort to make them secure still harbor covert channels introduced by both interfaces and specific code. It appears that an analysis of the same order of thoroughness as this must be performed at level B2 or higher.

Many of the fastest channels are CPU speed dependent. As technologies improve, these will only get faster. Better techniques are needed to eliminate, rather than just reduce their bandwidth.

---

[7] SVR4 is the base on which SVR4.1ES was built.

[8] IEEE Std 1003.1-1990, ISO Std 9945-1:1990.

ACKNOWLEDGEMENTS

Eric Lund, Jozef Chou, Kelvin Delvarre, and Ellen Fisher of USL performed the covert channel analysis on later internal releases of SVR4.1ES and supported evaluation by NCSC. They were also responsible for treatment of channels. Jeff Hostetler and John Johlas of Motorola MCG assisted in developing the CCA tool and in analysis of the kernel.

REFERENCES

[De76] Denning, D. E., "A Lattice Model of Secure Information Flow," *Communications of the ACM*, 19:5, May 1976.

[De77] Denning, D. E. and Denning, P. J., "Certification of Programs for Secure Information Flow," *Communications of the ACM*, 20:7, July 1977.

[Ke83] Kemmerer, R. A., "Shared Resource Matrix Methodology: A Practical Approach To Identifying Covert Channels", *ACM Transactions on Computer Systems*, 1:3, August 1983.

[Po91] Porras, P. A, Kemmerer, R. A., "Covert Flow Trees: A Technique for Identifying and Analyzing Covert Storage Channels", *IEEE Computer Society Symposium on Research in Security and Privacy*, May 1991.

[Ts87] Tsai, Chii-Ren, et. al., "A Formal Method for the Identification of Covert Storage Channels in Source Code", *IEEE Symposium on Security and Privacy*, April 1987.

TOWARD A MODEL OF SECURITY FOR A NETWORK OF COMPUTERS

William H. Murray
Deloitte & Touche
21 Locust Avenue, Suite 2D
New Canaan, Connecticut 06840
WHMurray@DOCKMASTER.NCSC.MIL

Patrick Farrell
Department of Computer Science
George Mason University
Fairfax, Virginia, 22030-4444
pfarrell@cs.gmu.edu

ABSTRACT

This paper proposes a model for evaluating the security of networks of computers. It is about the security of the collection of systems connected, rather than about security of the connecting infrastructure. It proposes that: 1) the networks of interest can be modelled as a collection of abstract single-user single-task machines connected at their user interfaces; 2) the security of such a net can be evaluated by comparing the cost of a successful attack to its value; 3) the cost of attack can be evaluated in terms of the work effort to the attacker and that the value can be evaluated by calculating the how much the success reduces the cost of subsequent attacks. Illustrations are given of how success against one node reduces the cost of attack against other nodes.

The paper suggests uses for the proposed model. It recommends network security practices that are suggested by the proposed model. It also suggests additional areas of research.

INTRODUCTION

To say that modern networks of computers are complex and dynamic is to restate the obvious. It is difficult to make statements or answer questions about any property of such networks; even such seemingly simple questions as size and scope resist easy answers. Nonetheless, mathematical modelling enables us to make useful, predictive, though not absolute, statements about even more complex systems, such as the economy or the weather.

PURPOSE

In a paper presented to the National Conference on Computing and Values, Murray[1] asserted that we need to be able to make statements about the security of populations and networks of computers as well as about individual computers. He suggested that we need to be able to answer questions such as what happens to the security of two systems when they are joined together? What happens to the security of a system when it is joined to a network? What happens to the security of the network?

The ideas presented here are intended to help us answer such questions. They are intended as a start toward a mathematical model of the security of a network of computers.

APPROACH

We begin with an abstract atomic system that is so simple that we can make statements about its security. This system is simpler than the kind of multi-user system that we normally use to illustrate computer security. It is intended to abstract a set of qualities and characteristics that will be relevant to its role in a network while putting aside such otherwise significant characteristics as application and environment which are obscured by its membership in a large population of dissimilar systems.

We are looking for properties that are fractal and composable, i.e., independent of scale, as true of the whole as they are of the parts. We are looking for properties that can be used to describe the security of

Copyright © 1992 William H. Murray & Patrick Farrell. All Rights Reserved.

otherwisc dissimilar systems. We are looking for properties that are independent of system type, management, or qualtiy of implmentation.

CONTRAST TO OTHER APPROACHES

We take note of Courtney's first law: "Nothing can be said about the security of a system except in the context of an application or environment." However, for our purposes, we assert that it is the generality and flexibility of the system that is of interest. For purposes of being able to talk about the network, we focus on those properties which are shared across systems, and ignore or conceal those that are unique or peculiar to a few.

This concept of security can be contrasted to that dealt with in the *Orange Book*[2]. The *Orange Book* wishes to be able to make absolute statements about single systems; we wish to be able to compare effects within populations of systems. The Orange Book describes levels of security in terms of policies and the ability to enforce them in terms of functions or capabilities. Our approach is more like that of the cryptographer, in which the assumption is that any code can be broken at some cost. We wish to be able to make statements about relative cost. That is, we wish to make statements about measures that will increase or decrease the cost of attack or the value of success.

CAUTION TO THE READER

This paper is directed at the collective security of computers networked together. It concentrates on the security of the collection as a whole, with recognition of the contribution towards collective security provided by each abstract machine. This paper is not about telecommunications or the connecting infrastructure (network), nor is it about computer security in the traditional sense. It is not about system security but about the security of networks of security.

THE SIMPLE MODEL

In this section we begin to introduce the primitive components of the model.

THE ABSTRACT MACHINE

Let M be an atomic single-user single-tasking single-service abstract machine of such strength and isolation that the only path of attack is through its user interface. M contains a secret, privilege, or resource, R, of value, V, and cost of loss or compromise, L.

Un-privileged user processes within a multi-user system would be modelled as a collection of such abstract machines. The privileged processes on such a machine would be modelled as containing, as resources, the entire collection.

SECURITY MODEL

Let C_a be the cost of an attack trial and N the expected number of trials required for success.

We define the abstract machine to be *attack secure* if the cost of a successful attack, C_aN is greater than V. That is, a rational attacker will not attack the system if the perceived cost of attack is greater than the value of success. As we show, this cost can be explicitly modeled.

Within successful attacks, we distinguish between *penetrations* and *breaks*. We would consider an abstract machine to be *penetrated* if the cost of attack fell below the value of successful attack and for the length of time that such condition persisted. We consider the machine to be *broken* if the condition would not remedy itself automatically. Thus, if the attacker were to obtain the password to a machine, the machine would be considered penetrated for the life of the password. If access to the machine included the

ability to change the password, and if the attacker changes it, then the machine is broken. The importance of this distinction can be illustrated by the following comparison. When my personal computer is in auto-answer mode, it is protected by a seven digit one-time password. It has a very high cost of attack. However, a successful attack includes the ability to insert a secret door in the one-time password mechanism, that is to break the machine. While it might require some special knowledge, and might be difficult to accomplish in a limited time, the privilege to do it is inherent in the penetration. The situation may not remedy itself automatically and the attacker can continue to use the system at reduced cost.

Contrast this to an unprivileged user ID on a Multics system using the same one-time password mechanism. A successful penetration, i.e., a one-in-ten-million guess, does not include the privilege of inserting a secret door. The abstract machine does not include the ability to alter its own logon security. Once the session is ended, the cost of attack returns to the old level automatically. The abstract machine was penetrated, but it was not broken.

While this definition of security may appear to be absolute, it is in fact, relative. We may never know all of the parameters of the model in the manner necessary to make absolute statements about the security of a particular machine. However, we may know enough to make comparative statements about two machines.

This scalar value can be replaced with a time series. Thus, the security evaluation term, $C_a N=V$ becomes $C_a(t_1)N(t_1)=V(t_1)$ at the time t_1. (While this is not important when evaluating the security of a single abstract machine, it will become extremely important when we attempt to evaluate a network.)

ASSUMPTIONS

We note the following simplifying assumptions:

- The model is limited to things that can happen at the user interface. The user interface is that which any user sees when he approaches the system. It includes all of the services that one can get with, or without, logging on. It includes even those services that are normally reserved to privileged users if they are available a the same interface. We ignore things that might happen at a privileged or local interface that would not be visible from the network.
- An attack is an attempt to obtain any service to which one is not normally authorized. It will usually include one or more attempts to logon to the system. Each trial, or associated group of trials may be modelled, examined, or evaluated independently.
- R is the target of attack, the product of successful attack, and that which the owner desires to protect. R may be a secret, such as intelligence or a password; a privilege or a capability, such as the ability to modify a program or place a paid phone call; it may be a connection or a path to a nearby system; or it may be computing capacity, such as might be used to attack a nearby system. For our purposes it is monolithic. Since M is single state, then if the attacker has any part of R, he is assumed to have all of it.
- V and L are related but not necessarily equal. The value of R to an attacker may not be the same as the cost of loss to its owner.
- The cost of access to logon is assumed to be part of the cost of attack, constant, and the same for all potential attackers. Since this is not true in the real world, the model will have to be elaborated to show the components of this cost, and its distribution among attackers.
- The number of trials for success, e.g., half the size of the logon space, is known and constant; again, the model can be elaborated to adjust for uncertainty here. Likewise, the size of the space can be increased, within bounds, to compensate for increases in the value of V. However, all other things being equal, the bounds on the size of the space will ultimately impose an upper bound on the value

of V. That is to say, since there are upper bounds to the size of a password, there are effective bounds on the size of V.

- The number of targets is limited to the single system, M, with value, V. This limit will be relaxed by dividing M, adding it to a population, or by joining it to other systems to form a network.
- We measure work in only in time. The value that the attacker places on his time is not equal for all attackers and not available to us. Neither is it necessary for answering the questions that we want to answer.
- The choice of a single user system is used for simplification. The model will be elaborated to account for the effect of multiple users
- A single task machine was chosen to make logon the only security mechanism of relevance. That is, a compromise of logon will result in a total loss of V; no other compromise is necessary. This choice was made to illustrate the relationship between the cost of attack and the resource. The elaboration of the model will have to account for multi-user systems with multiple alternative paths of attack to R, and for more complex costs of attack. Conversely, the model will have to be able to account for protective security based upon the division of R across compartments within M.

These assumptions and constraints illustrate some of the dimensions that must be included in a fully elaboration of the model.

THE COST OF ATTACK

The model of the cost of attack will be modelled, in part, by modelling the cost of individual trials. This cost will include the cost to send a unit of coded data times the number of units in the trial, plus the time required to receive and evaluate the response. Thus, for example, the cost of sending six characters is higher than that of sending five, the time required to send it at 300 baud greater than that of sending it at 2400 baud.

We have defined the cost of attack as the cost of a trial times the average number of trials required for success. In its simplest form, $C_a := C_t\, E[n_a]$.

The cost of a trial is a function of the resources available to the attacker, knowledge, coded information, and capacity. It is related to the protective measures in place. That is, a change in the protective measures may alter either the cost of the trial or the expectation of success of the trial.

If the probability of the success of a trial were to remain constant for the duration of the attack, then a simple Bernouilli distribution could be used to calculate the expected value of the number of attempts required.

$$E[n_a] \equiv \sum_{i=0}^{\infty} i\,(1-p)^{i-1} p = \frac{1}{p}$$

This is the same formula used in telecommunications analysis to the calculate the expected number of retransmissions for a given error probability.

However, while we speak of the average number of trials for success, there may be a difference in the cost of successive trials. Even a trial that does not yield success may yield information that lowers the cost or improves the chances of success for later trials. For example, in an exhaustive attack against a password, each trial reduces the size of the remaining space and improves the probability of success of the next trial.

In addition, protective measures may not operate the same on all trials. For example, modern systems include a number of measures that are intended to raise the cost of sucessive trials. These include:

- delay the prompt after a failed trial, reducing the effective bandwidth and increasing the cost of successive trials[4].

- set an alert threshold at a specific number of failed logon attempts for a given userid and then revoking that userid; and
- break the connection after a threshold number of failed attempts, thus adding the cost and delay of a re-connection to the next trial.

Note that these techniques are not intended as an impregnable defense. Rather they are intended to raise the cost of attack. However, in a target-rich population they succeed in moving the attacker to other targets with lower cost of attack.

The model must also account for special knowledge or experience available to the attacker. For example, attackers know that some passwords are far more likely than others. In the population attacked by the Wank worm, one system in five had at least one password equal to the null password or to the user ID. Short dictionary (or "sweet list") attacks are a staple[5]. In order to extend the model to reflect the variable costs, we first consider the cost of the attack, which is equal to the sum of the individual trials, over the expected number of attacks:

$$c_a = \sum_{i=0}^{E[n_a]} C_i$$

To calculate each C_i value, we need to consider how the above cost increasing techniques can be modelled. As a simplifying assumption we will consider that the primary cost associated with an attack is driven by the time required for the attack. This is appropriate, for example, if the attacker had to pay for a long distance telephone call during the attack. It is also appropriate if the chance of detection is proportional to the duration of the attack.

A realistic cost formula can be formulated from the following variables:

$K_{\Delta t}$	:=	linear cost factor
MAAT	:=	Mean active attack time. This includs the number of characters required to send a userid/password pair to the abstract machine's user interface, divided by the network bandwidth.
FailDelay	:=	Time delay inserted after a failed access.
NumTries	:=	Number of tries allowed before the communications link is severed.
ReconnectTime	:=	Delay cost associated with connecting/reconnecting to the machine.
KillThreshold	:=	number of attempts allows before account is invalidated.
i	:=	Attack iteration number.

$$C_i = \begin{cases} K_{\Delta t}\left[(MAAT + FailDelay) + (\text{if}(i \bmod NumTries = 0)\,\text{then}\,ReconnectTime\,\text{else}\,0)\right] & \text{if } i < KillThreshold \\ \infty & \text{if } i \geq KillThreshold \end{cases}$$

Clearly if the expected number of attacks, $E[n_a]$, is greater than the *KillThreshold,* the individual C_i terms become infinite, and the summation also becomes infinite. For those cases that are interesting, this yields:

$$E[C] = K_{\Delta t}\sum_{i=1}^{E[n_a]}(MAAT + FailDelay) + K_{\Delta t}\sum_{i=1}^{\frac{E[n_a]}{NumTries}}\text{Re}connectTime$$

$$E[C] = E[n_a]K_{\Delta t}\left((MAAT + FailDelay) + \frac{\text{Re}connectTime}{NumTries}\right)$$

THE VALUE OF SUCCESS

A successful attack gives the attacker access to whatever capacity, paths, abstractions and recorded data that are implemented in the target. For purposes of the model it is necessary to divide these resources into those which reduce the cost of attack against the network and those which do not. While the latter may be

intrinsically valuable and influence the attackers propensity to attack this particular machine, only the former influences the security of the network.

Capacity

Capacity is the most fungible and easy to measure and appreciate of the values in the target. Until consumed, i.e., used or wasted, it can be devoted to any purpose. For example, it can be used to reduce the cost of attack against another system.

Identity of the Victim

The successful attacker obtains the identity of the victim node. That is, he gains the ability to appear as that node to other nodes in the network. While under the compromised identity, the attacker is accorded all the trust and tolerance that would be accorded to the victim.

At a minimum, this is the tolerance that would be accorded to any member of the network or community. Historically the network has been both orderly and trusting. Most of the behavior has been orderly. In part because of this and in part because the network was new and small, users have been tolerant of eccentric behavior on the part of other users. They have also been accomodating and helpful to other members of the community. Therefore, simply being able to act under the identity of a legitimate member of the community is helpful to an attacker.

At the most, the attacker may gain all of the privileges and trust accorded to the most privileged and trusted administrator.

Identity of the Information

The amount of information that the attacker gets from coded data is, in part, a function of what he already knows. While a bit is the amount of information that reduces uncertainty by half, the recorded code of that bit is valuable only if one is able to recognize, i.e., identify, the bit.

Currency of the Information

The value of most data will go down with age. For example, even reusable passwords may have a finite life. The later in its life it is learned, the lower its value. This leads to at least three alternative models for information currency:

- Linear decay where the value of the information decays at a constant (and potentially zero) rate;
- Exponential decay, perhaps using a half-life model or other exponential decay; and
- Abrupt expiration, where at a specific point, the value drops instantly to zero.

Combinations of two or more of these models may be suitable for specific real world problems. For example, a userid/password pair that will expire at a fixed point in the future provides a capacity value that linearly decreases. Exponential decay is suitable for information that loses relevance over time, such as a data store that contains the network topology as of a given date in the past. Since the network is continually revised, the information becomes outdated, the general information in the data store will remain valuable after the detailed information is obsolete.

Scope or Quantity

All other things being equal, the more data, or the wider its scope, the more value. The value of data will normally rise along an "S" shaped curve with quantity. Thus, the more other systems a given target knows about, the greater the value of a successful attack against it. The initial, small amounts of information do not yield much knowledge (or value). As a critical mass of information is gathered, each datum provides an

incremental increase in value. When nearly all the information has been captured, the incremental benefit tapers off. Thus the more other systems a given target knows, the greater the value of a successful attack against it.

Development of a formula that maps a linear estimate of information scope onto the "S" curve is easy. The difficult part is development of a quantitative metric that maps a given amount of information to an estimate of the percentage of the abstract machine's total information.

MODELLING THE VALUE OF SUCCESSFUL ATTACK

In modelling the value of successful attack, we make three useful distinctions. First we distinguish that part of V that is useful in directly lowering the cost of attack against the network from any other resources. For example, information about the network or other machines will lower the cost of attack against thoses machines, while otherwise very valuable financial information will not. This latter data may have intrinsic value which adds to the attractiveness as a target of the machine, but does not impact the security of the network.

Second, we distinguish between that portion of V which can be modelled as another abstract machine and that which cannot. For example, if access to machine A includes cheaper access to B and C than is available without access to A, then we would model that access as abstract machines with the new, lower, cost of attack.

Third, we distinguish between those resources whose identity is likely to be known outside the abstract machine. For example, if an abstract machine is an instance of a population of similar machines, then identities which it shares with those other machines will be widely known and exploitable. For example, information in a TCP/IP "host table" would be readily identifiable and exploitable, as it was in the case of the Internet worm.

MODELLING A NETWORK

In this section we define a network, extend the model, and apply it to the network as defined. The definintion of network that we suggest is special, but it does include the networks of interest.

DEFINITION OF A NETWORK

We define a network as two or more of our abstract machines connected in such a way that the users of one can see the user interface of the others, or an aggregation of such networks so connected. We deal with the collection of things connected, rather than with the connecting infrastructure. We deal with networks of computers, not networked computing.

For the moment and for our purposes, we define the network to be so physically secure, protected, or isolated that the only viable attack is via the user interfaces. By this means we exclude from the model the issues of eavesdropping on the media, violence, subornation, or coercion. We assert that these issues are general to remote computing of any kind, rather than peculiar to networking of computers. We recognize that there are some among our possible readers who believe these to be the most interesting issues. We beg their indulgence and patience. We suspect that by binding some other variables in the model constant, it can later be used to examine these interesting issues.

This definition would not treat a single, abstract machine as a network. It would distinguish between a network and a collection of such abstract machines based upon whether or not the machines were connected in such a way that the user of one machine could connect to and exploit the user interface of another. Based upon this distinction, most multi-user machines would not qualify as networks, since one process is not normally able to connect to another without its knowledge and cooperation. On the other hand, two connected single user systems might qualify. Two connected multi-user systems might qualify as a complex network.

For the moment, and as a working hypothesis, we assert that for the purposes of our model, the most significant difference between whether these machines are two boxes connected by wire or are compartments in a single box is one of bandwidth. That is, all other things being equal, the link between two compartments in the same box will support more attack trials per unit time than will a link between two separate boxes. but that no other difference is important to our purpose.

For purposes of this work, it is essential that this definition preserve the significant properties of our abstract machine and model or reflect the security propeties of networks of real systems. Our model is subject to examination and criticism to ensure that it does so. Nonetheless, it is in the nature of a model to emphasize some properties of that being modelled over others.

DEFINITION OF NETWORK SECURITY

As with the single abstract machine, such a network is defined as being attack secure if the value of a successful attack is less than the cost of such an attack.

Note that such a level of security does not necessarily prevent all attacks. It does not make the network resistant to all attacks. It does not even prevent successful attacks against some systems in the network. On the other hand, neither does it require that all components of the system be attack secure in order to say that the network is secure.

The analogy here is to a fishnet rather than to a chain. While each is weakened by the compromise of a link, the net can continue to work in the face of many broken threads or knots. Once more, the purpose of the model is to ba able to talk about relative, rather than absolute, security.

AN ILLUSTRATION

For purposes of clarification it may be useful to apply our model to some existing systems and attacks. Consider first one of the Unix systems that was attacked by the Internet worm[3]. Since those machines were not atomic machines, they were, for our purposes, either a collection of abstract machines, or a network of such machines. That is to say, `sendmail`, `fingerd`, `debug`, and each of the user processes would each be modelled as a separate abstract machine. They would not be modelled as a network because they were not connected in such a way that the user of one such process could see the user interface of another from that process. Debug would be modelled as an abstract machine within a machine, sendmail. On the other hand, any of these abstract machines would be modelled as a network with user processes in other Unix machines. The value of a successful attack against debug includes access to all of the processes within the Unix machine.

Note that the real cost of attack in this system was measured in terms of tens to low hundreds of man hours. This was obviously and certainly lower than the cost of the loss. Thus, the network was not protection secure. While no value was converted to the attacker, it is possible that a small variation on the attack might have yielded considerable resource. Therefore, by our definition the network was not attack secure. Note also that the penetration of one system lowered the cost of attack against subsequent ones. Once the worm succeeded in getting itself executed in a target machine, it gained a path to additional machines, knowledge about them, and capacity with which to attack them, i.e., it could and did start attack processes against them.

EFFECT OF NETWORKING ON SECURITY

We assert that, all other things remaining equal, the relative security of a system goes down when it connects to any other system. That is, the population of attackers with connection goes up, the cost of attack goes down, and the value of success goes up. The cost of attack goes down because there is more capacity to be used in the attack. The value of success goes up because the value included in the target goes up by the value of the connection to other targets.

SPATIAL RELATIONSHIP BETWEEN ABSTRACT MACHINES ON THE NETWORK

We assert that within the context of this model of abstract machines, the proper model is a fishnet, not the more common model of a chain. Specifically, a fishnet may have large holes, and may even have a number of large holes, while still providing its design function. With a fishnet, some holes cause problems, and others do not. This model of network security shares this characteristic.

As in a fishnet, the potential harm caused by a specific hole is related to the proximity of this weakness to others. Additionally, we suggest that the value of an attack is decreased with increasing distance. Many mathematical models are available that can show this decrease in value. We suggest that the inverse square law, that is often encountered in physical phenomina such as light or other electromagnetic emmisions, is a suitable and well understood description for the decreaing value of an attack over distance. We propose that rather than a traditional geometric distance metric, an alternative metrics for distance are either "time" or "work." These terms are interdependent; either may be used to suit the specifics of the model under study. The amount of effort required to set up for an attack, or the exposure and thus the probability of detection during an attack, measures the cost to the attacker, and is used as the distance metric.

USES OF THE MODEL

We think that the ideas and abastractions presented in this paper have a number of useful applications. First, we submit that the simple abstract machine and simple security model provide a useful way to think about security, i.e., relative rather than absolute. Second, modeling a network as a collection of simple abstract machines enables us to isolate and examine properties of the network which are important to its security. Finally, they can be used to support the additional research which we recommend below.

SUGGESTED RESEARCH

This paper is entitled "Toward a Model...." That is intended to suggest that we do not think or suggest that this paper represents completed work. We suggest three paths.

First, as with most such models, this one can be elaborated in both detail and scope. For example, more components of the cost of attack can be identified and described. More important, the results of success can be identified, classified, evaluated, and described. The object would be to determine the resources that are of use to an attacker, how he might use them, how much they will reduce his cost of attack, and how that value to him may be limited or reduced. All of this needs to be rigorously and completely described mathematically.

Second, a simulation can be constructed based upon the model. We believe that the a network simulator can be built based upon the properties of the network components and the definition of security that we have described. This simulator would enable us to assess how the security of the network responds across time to attack.

Third, experiments can be conducted using the simulator. For example, a network of given topology, cost of attack, and value of success can be simulated. The simulator can be used to evaluate the resistance of the network to varying kinds of attacks or varying resources available to the attacker. By holding the form and resource of the attack constant, and varying the topology of the network, the experimenter will be able to compare the resistance to attack of different topologies. By varying the Logon mechanisms of various nodes, the evaluator will be able to measure the cost to the attacker that results from different kinds of security mechanisms and determine the effect of their distribution within the net. That is, one could assess whether the resistance of the network to attack can be raised by raising the cost of attacking only some of the nodes. By holding everything else constant, and varying the privileges and information about other nodes in the abstract machines the simulator can be used to measure the effect of such measures as compartmenting privileges or encrypting password tables.

RECOMMENDED PRACTICES

This analysis suggests a number of practices that can improve the security of networks of computers. While raising the cost of attack these measures will have a minimum impact on legitimate users acting in the intended way. These practices include:

- Limit the life passwords; prefer one-time passwords for most uses in networks.
- Consider dual passwords for privileged accounts.
- Require cooperation of two people to alter logon programs.
- Automatically reduce the bandwidth in the face of possible attacks (delaying the prompt is the prefered mechanism because it corrects itself automatically; alternatively break the connection, revoke the user ID, or otherwise disable the logon);
- Do not offer any services or information to unknown or untrusted systems or users. Offer GUEST and ANONYMOUS services only from systems specifically intended for that purpose.
- Limit the information that is given to unauthenticated users. Specifically, do not grant access to logon IDs, do not tell the user what is wrong with a logon attempt, and do not tell them that an attack is suspected. Do not tell them the identity or type of the system or network.
- For security, compartment the network into subnets, name spaces, and domains that each require there own authentication.
- Prefer multiple user names and authenticators for sensitive applications; use single name spaces for user convenience.

REFERENCES

[1] Murray, William H., *On Computer Security and Public Trust,* Proceedings of the National Conference on Computing and Values; August 1992, New Haven, CT.

[2] Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria, National Computer Security Center, Ft. Meade, MD, 1987

[3] Spafford, Eugene H., *The Internet Worm Program: An Analysis*; ACM Computer Communication Review; January 1989; Number 19(1).

[4] Capek, Peter, *Discouraging Penetration Attempts on Interactive Computer Systyems Without Denial-of-Service,* IBM Technical Disclosure Bulletin, February 1989, Volume 31, Number 9, Page 147-149.

[5] Stoll, Clifford, *The Cuckoo's Egg,* Doubleday 1990, New York, NY.

Towards a Policy-Free Protocol Supporting a Secure X Window System

Mark Smith

AT&T Bell Laboratories
Greensboro, NC 27420-0046

Abstract

Keywords: Graphical User Interfaces, X Window System, interoperability, multi level secure.

This paper proposes a framework for a secure, interoperable X Window System[1] protocol. It reintroduces the concept of a *policy-free* protocol within the context of the X Window System with the goal of achieving industry consensus on that protocol for secure operation. We claim that this consensus can be achieved without requiring vendors to agree on a single standard security policy, much less agreeing on a particular implementation of a security policy.

The policy-free protocol framework and its impact on X Window applications will be proposed and discussed in some detail. The paper will also discuss the relevance of this approach to other trusted systems.

1. Introduction

The problem of constructing a secure X Window system has been treated in several prior accounts [2, 4, 6, 11]. Epstein [4] in particular contains a breakdown of the secure X Window problem by problem area (e.g. mandatory access control, discretionary access control, object reuse) and by level of trust (e.g. B1 trust, Compartmented Mode Workstation trust, B3 trust). These treatments hint at the problem of creating a *secure interoperable* X. In order for any X Window system to be consistent with X's original design goals, it must be interoperable. That is, it must be possible for the user to run an *X Window client* from any machine on the network where the X Server runs, independent of the hardware architecture of the machine the client is running on. See Figure 1 for a possible X Window topology.

Interoperability is attained through the specification of a standard X protocol as defined by the MIT X Consortium. As vendors gain more experience with X and desire additional X capabilities, the standard evolves. There is a facility called *extension* that allows a vendor to "burn in" novel X protocol extensions. These vendor-specific extensions may be reviewed by the MIT X Consortium for eventual inclusion into a new X standard. When an extension is approved and included in the X Window standard, the facilities it provides become effectively interoperable. The MIT X Consortium also includes in its "sample server" a set of extensions which are not yet in the standard but which are deemed to be of sufficient value to warrant inclusion. A vendor need not (and sometimes does not) pass on the MIT sample server verbatim to its customers. Instead, a vendor may choose to select or reject any extensions it receives from the MIT sample server. An extension thus becomes effectively interoperable if all vendors choose to include it in their delivered X Servers.

There are two problems with X Window security extensions:

1. The MIT X Consortium has historically included only the most minimal notion of access control in its X protocol standard.
2. Several vendors have added access control extensions to the X protocol. These extensions are not interoperable; moreover, they do not necessarily reflect the same security policy.

We propose a framework for the creation of a single X protocol extension that is capable of supporting all the security features, attributes and policies that vendors (and their customers) desire. Industry consensus on a protocol fitting this framework would yield an X Window extension, leading eventually to an interoperable secure X Window system.

1. The X Window System is a trademark of the Massachusetts Institute of Technology.

The proposal defines separable *policy-free* and *policy-defining* subsystems. This rearchitecture of security-providing facilities is similar to that proposed in [1], [3] and suggested by [12]. In particular, [1] suggests a policy-free mechanism for access control with architectural advantages similar to those of our proposal. Perhaps more relevant is the fact that X was originally designed based on the principle of a policy-free protocol [7]. X is *window management policy-free*, allowing vendors the freedom to design and develop X window managers however they see fit. Though there has been some criticism of policy-free protocols for use in a graphical system, they have the distinct advantage of allowing vendors to standardize on relatively non-controversial, mechanistic protocols, rather than on much more controversial window management policies, for example.

Experience has shown that similar difficulties exist when vendors attempt to standardize on a particular protocol supporting security attributes or policies. Even in cases where vendors agree on the security policy (e.g. the Compartmented Mode Workstation policy), they do not necessarily agree upon the protocol, since security policies can be implemented using various techniques (e.g. separation vs. fine grained access control) and typically these techniques impact the protocol. Furthermore, certain aspects of "standard" security policies may be left to the interpretation of the vendor and the accrediting body, and different "subpolicies" may be allowable under the standard (for example, access control lists are optional for the Compartmented Mode Workstation). Finally, a single vendor may desire to support a customer base demanding different security policies.

What is desired therefore is a protocol that passes two tests: (1) it must allow the vendor to use whatever technique the vendor desires to implement the security policy, and (2) it must allow implementation of the security policy of the vendor's choice. It is also highly desirable that the protocol minimally impact the performance of the system. We claim that only a protocol passing these tests has a chance to attain vendor consensus leading to interoperability.

A framework for creating such a policy-free protocol will be proposed in the following section. The protocol's impact on the X Server and policy-defining X clients will then be explored, followed by a summary of the expected impact on the embedded base of X Window clients (referred to as the COTS, or commercial off the shelf, client base). Finally, a generalization of the architectural principle of policy-free interfaces supporting security policies will be briefly discussed.

Throughout this paper, the term "X" will refer to the MIT X Window System, "CMW" will refer to the Compartmented Mode Workstation, and "ACI" will refer to Access Control Information.

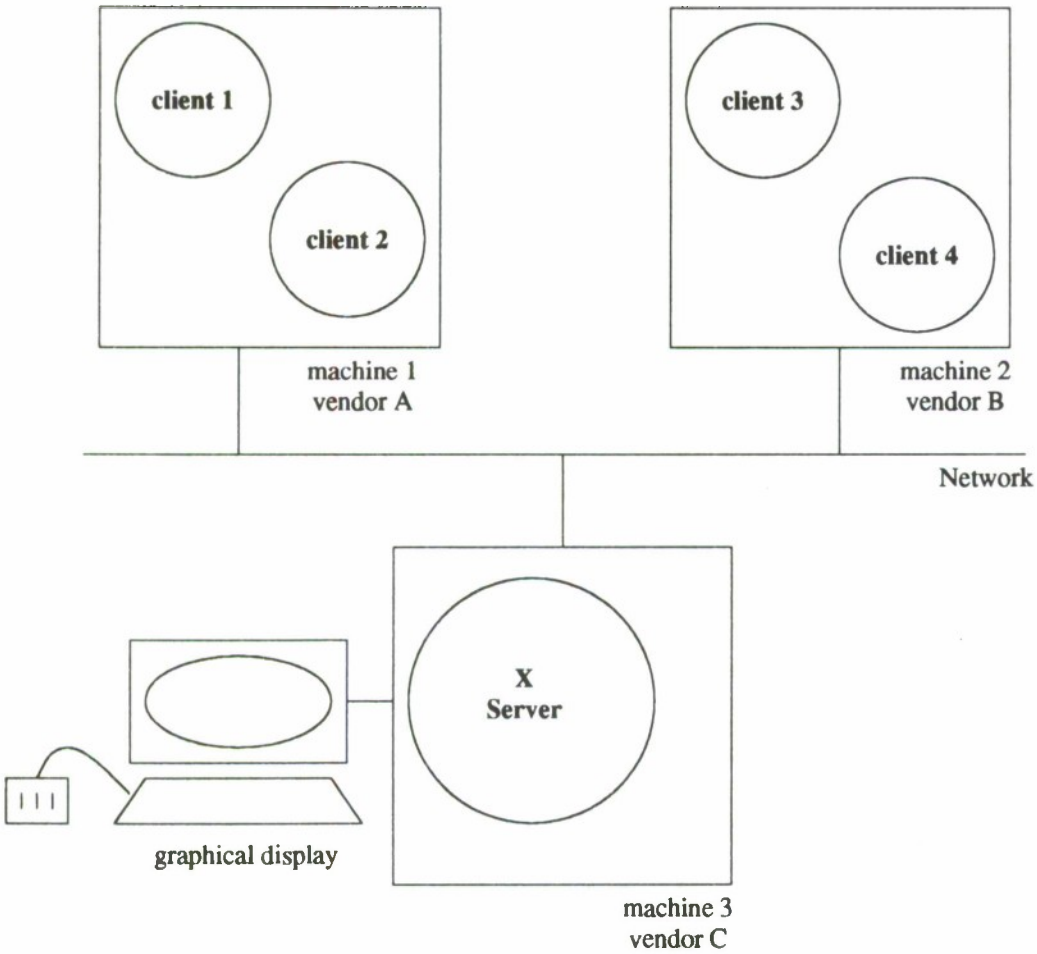


Figure 1. An X Window topology

2. Proposed Protocol Framework

2.1 Protocol Overview

The attempt to specify a protocol that supports various security policies and implementations thereof is similar to the functional decomposition methodology of an object-oriented system, or more particularly that of a strongly typed system. This decomposition is thus similar to the one described by LOCK [10], which specifies security policy based on object and subject type rules.

In particular, "subjects" and "objects" should be type-classified in a policy-free protocol in order to support a large class of labeling security policies. However, such a classification is not sufficient to specify very fine-grained policies that make use of object access control lists, or subject capabilities [12]. Therefore, for the protocol to be sufficiently general, it must also include the ability to identify ACI on a per-subject or per-object basis.

There are other issues to consider besides what might be called "pure" access control. The protocol must not preclude the construction of a secure infrastructure capable of being accredited. Such an infrastructure must pass certain integrity tests; for example, a change to an object's ACI cannot be made while it is being accessed (the *tranquility* property). Also, provision must be made for trusted paths, so that X clients doing trusted I/O have provably unspoofable access to the physical display, and so that the X Server has unspoofable access to trusted policy-cognizant X clients. Other infrastructure requirements will be introduced in the following sections as well.

2.2 Architecture

The key aspect of the architecture (Figure 2) is the construction of a *policy-defining client*, which we will abbreviate PDC. It is this client which provides the basic Access Control Decision Function (ADF) as defined in [3] and [12]; the X Server, in nearly all cases, provides the analogous Access Control Enforcement Function (AEF). Note that the PDC has the same client/server relationship to the X Server as any other client; it can be similarly relocated to any machine on the network.

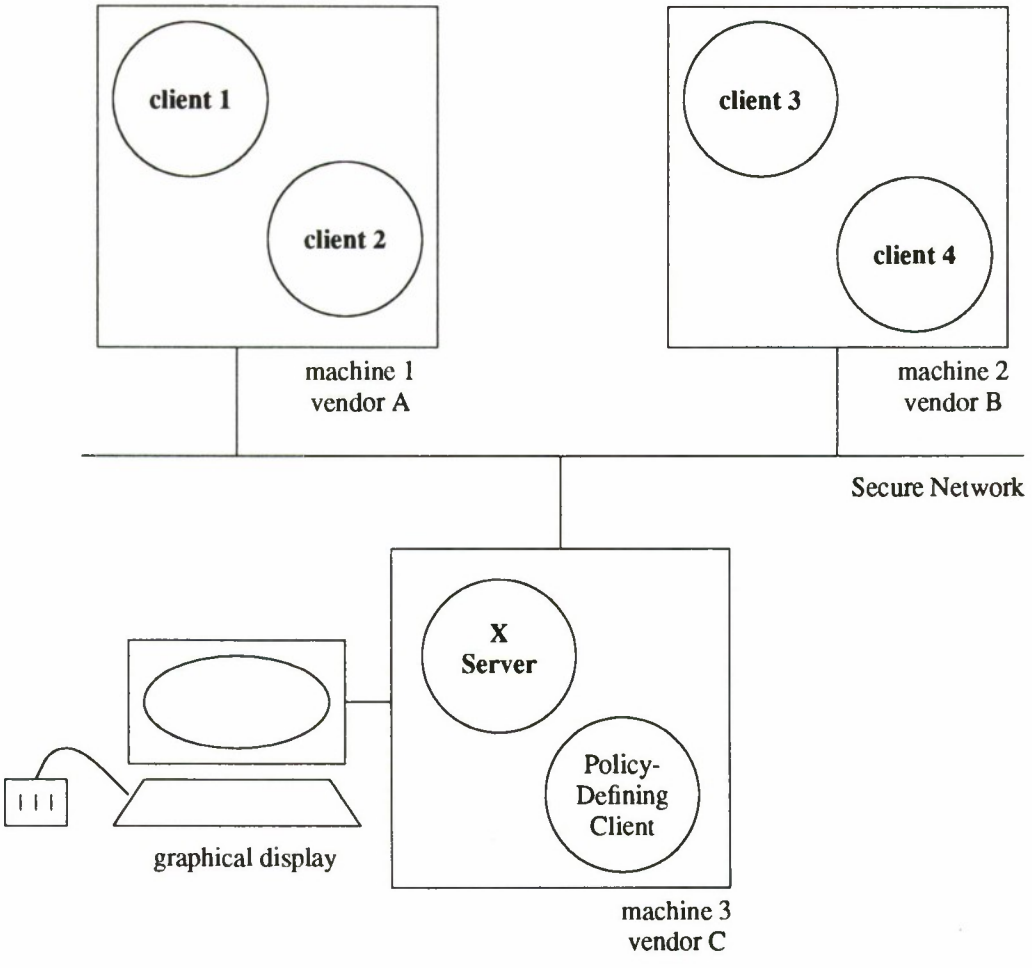


Figure 2. An X Window topology with a policy-defining client

In addition to the server/client relationship, the network over which the X Server, PDC, and other X clients communicate must provide certain authentication capabilities. In particular, it must not allow an untrusted application to spoof the PDC, and it must allow the PDC to authenticate any other client. This particular requirement also implies that the traditional meaning of "X client" must be strengthened: historically, "X client" has meant "a connection to the X Server." In theory, the X protocol allows more than one process to communicate on a single connection to the X Server, although in practice this capability has not been widely used. In order for the PDC to authenticate a client, it must assume that a client is a subject, and therefore there can only be a single identifiable client on a connection. The network must provide a way to restrict access to a connection in the required manner.

2.3 Proposed Mechanisms

The framework for the policy-free protocol will be summarized by high-level functions enumerated below. These functions can be considered at the logical level of the Xlib interface [8], and similarly, each one would correspond to a particular protocol request or event.[2]

2.3.1 Access control mechanisms (including cut and paste) The proposed framework implies that the X Server assume the role of Access Control Enforcement Function (AEF) and that the PDC be the Access Control Decision Function (ADF). Basically this means that the X Server sends an event when a subject attempts to access an object, and that the PDC replies with a request indicating (1) whether the access was granted, and (2) if not, what error condition is represented by the access denial.

This separation of duty allows a clean implementation of *mediated* policies. The most well-known of this class of policies is the CMW cut and paste policy. In this policy, when a subject attempts to paste previously cut data into a target window whose security classification differs from that of the source window, an interactive window session is required before the paste operation can complete. This interaction reminds the user of the source and target window classifications and asks the user to verify the reclassification. Such an interaction can be handled cleanly and without race conditions if the PDC initiates the interactive verification session upon receiving the appropriate access control request. Other interesting mediation policies can be similarly implemented.

The separation of duty also allows a straightforward implementation of floating information labels as specified by the CMW requirements. In particular, the PDC can implement the policy whereby a successful "read" access to a particular object by a particular subject results in a change to the ACI of the subject, based on the relationship between the client's current ACI and the newly accessed object's ACI.

The following mechanisms are sufficient to implement access control:

- **RequestAccess(entityID, entityType, clientID, accessMethod)** event. The X Server sends this "event" (really a request) to the PDC whenever a subject (specified by **clientID**) attempts to access an entity[3] (specified by **entityID**). The X Server also sends the entity's type (**entityType**).

  Entities in the X Server must be classified by type in order to take advantage of type-based policies. The decomposition of X Windows entities into types has been treated elsewhere (e.g. [2]) and is outside the scope of this paper. We generalize the usual object decomposition somewhat by allowing the definition of subject access policies. This is necessary in order to implement a generalized form of privilege described later.

- **AccessAnswer(yesno)** request. This "request" (really a reply) is the PDC's reply to the **RequestAccess()** policy question. **yesno** contains the answer, either 'yes' or an error return defining what the access denial policy is for this particular access request.

Note that the PDC need not send an immediate **AccessAnswer()** reply upon receipt of a **RequestAccess()**. Rather, as described above, it can implement a mediation policy on the access, or even delay the request to throttle a covert channel, before replying.

A **RequestAccess()** might also signal the PDC to create a new access control binding. For example, the successful attempt to create a new object would typically cause the PDC to synthesize new ACI (perhaps from the subject's ACI) to be bound to the object.

2.3.2 Access control information binding mechanisms There must be a way for X to bind ACI to subjects and objects. In order for the protocol to remain policy-free, this ACI must be uninterpreted by the X Server. The protocol should therefore provide only a transport mechanism and binding semantics for ACI.

---

2. A *request* is a protocol message from an X client to the X Server; an *event* or *response* is a protocol message from the X Server to an X client. The X protocol is asynchronous; the result of a failed request is typically an error response. Note that a protocol message is classified as a request or as an event based solely on whether it is input to or output from the X Server.

3. An *entity* is a subject or an object.

An important consideration is the location of the bindings between entities and ACI. These bindings should be stored in the PDC, since it must be able to make access decisions based on current bindings. If the X Server stored the bindings, it would have to verify the validity of every **AccessAnswer()** request by checking the existence of the accessed object.

The ACI binding may be rather static (e.g. Bell-LaPadula mandatory access) or may be rather dynamic (e.g. CMW subject floating information labels or object ACLs). The potential for dynamism requires that the binding mechanisms be as general as possible, allowing the PDC fine-grained control over the ACI.

The following mechanisms are sufficient to implement ACI binding policies:

- **BindClientACI(clientID, clientACI)** event. The X Server sends this event to the PDC. The event contains the handle of a client (a **clientID**) that just connected to the X Server, and the ACI of this client as reported *from the network*. Thus, a network supporting a secure X Window system must be able to provide this service; historically, similar services have been proposed for secure network services such as MAXSIX [13][4]. The PDC binds the **clientACI** to the client denoted by **clientID**[5].

- **BindObjectACI(objectID, objectACI)** event. This event is similar to **BindClientACI()** and requires that the PDC bind the access control information to a particular X Server object. It is expected that this event is the result of a privileged client's desire to change the ACI of a particular object (see "Policy emulation mechanisms" section below).

  The PDC is also capable of binding object ACI unprompted; e.g. as a side effect of a successful **RequestAccess()** event requesting that a new object be created. In that case, a reasonable policy would be that the newly created object is bound to ACI derived from the creating client's bound ACI.

2.3.3 Privilege assertion mechanisms There are two general classes of privilege which are relevant to the protocol framework. First, there are subject privileges which are used by the PDC in order for it to implement windowing security policy. Many vendors desire to define fine-grained privilege policies which allow clients the right to enable and disable their own current privilege set (*privilege bracketing*: see [4]). The following mechanism is sufficient to allow *in-band* (that is, X protocol) privilege bracketing:

- **BindClientACI(clientACI)** request. This request has syntax similar to the **BindClientACI()** event described above. In this case, the client requests that its privilege set be changed as specified by the **clientACI**. The X Server forwards this request to the PDC using the **BindClientACI()** event. The PDC defines a policy which decides whether to honor the ACI (privilege) change request, presumably based on the ACI already bound to the requesting client.

Note that it is possible for the PDC to grant fine-grained policy-defining privileges to other clients as it sees fit using only the mechanisms supporting this first class of privilege[6].

---

4. In general, any secure distributed system must include a networked identification and authentication service; otherwise a remote trusted server cannot enforce policy. A formal definition of this network service is not an X interoperability issue and is beyond the scope of this paper.

5. The format of the client ACI should be general enough to handle all types of access control information in a machine-independent fashion. The formal specification of this format should be defined by a standards body and will not be further explored here.

6. One possible use of this mechanism would be to support a complex privilege-bracketed information labeling scheme. For example, a vendor may wish to implement a policy whereby the *xterm* terminal emulator will change the visible information label in a window based on the information in that window. One way to implement this policy is as follows: (1) *xterm* is granted the "create TCB-private window" and "change privilege to write to TCB-private window" privileges by the secure OS. (2) The *xterm* is bound to these privileges when it connects to the X Server. (3) *xterm* creates a TCB-private label window, which the PDC allows. (4) When it notices that it must write an information label, *xterm* requests that it be given the right to write into the TCB-private area. The PDC grants the privilege. (5) *xterm* writes the new information label into the TCB-private window, which the PDC allows. (6) *xterm* relinquishes its right to write into the TCB-private window. It is worth noting that the X Server was never aware of the semantics of either of the privileges used in this scenario.

The second class of privilege is the class of policy-defining privileges, in particular, the privilege to be a policy-defining client. A mechanism must exist allowing the PDC to declare itself as a PDC to the X Server. The following mechanism is sufficient to support the policy-defining privilege:

- **AssertPrivilege()** request. This request simply informs the X Server that the requesting client wishes to be the PDC. The first client requesting the privilege is granted the privilege. If the request is not granted, the requesting client receives a failure notification.

It is required that the underlying secure operating system provide a trusted path and trusted startup semantics so that the PDC is guaranteed to be the first X client to send this request. (One possible implementation of this trusted path would be for the UNIX[7] *login* program to start up the PDC, which would in turn start up the X Server and send it the **AssertPrivilege()** request. Should a spoofing PDC intercede, the trusted PDC would receive a failure notification and would be able to terminate the session and audit the spoof. Since the real PDC is using an underlying network trusted path, it can trust that the reply from the **AssertPrivilege()** is genuine.)

2.3.4 Policy cognizance mechanisms There needs to exist mechanisms whereby a client can be *cognizant* of the policy that PDC defines. An example of such a client would be a *gadget manager*, where a *gadget* is an object defined by the client (and whose type is not known by the X Server). A client may define a gadget such that it emulates an X Window, for example. Such a client would need to know what window policy is being defined by the PDC in order for it to emulate the same policy for the objects it defines.

The mechanisms to support this cognizance capability are basically extensions of mechanisms already defined. These mechanisms are:

- **RequestAccess(entityType, entityACI, clientACI, accessMethod)** request. This request has syntax similar to the **RequestAccess()** event described previously. In this case, however, the policy cognizant client sends the request to the X Server, which forwards it to the PDC. The PDC then sends the reply back to the X Server as defined above. Note that the policy cognizant client must specify the type and ACI of the accessed entity, and the type of the accessing client, in order to find out what policy the PDC enforces.
- **AccessAnswer(yesno)** event. When the X Server receives the answer from the PDC, it forwards it back to the emulating client as an event.
- **GetACI(entityID)** request. The policy cognizant client requests that the ACI bound to **entityID** be returned to it. The X Server forwards the request to the PDC. It is expected that the PDC will require that the requesting client pass access control checks before it returns the bound ACI.
- **BoundACI(entityACI)** event. The **entityACI** is returned to the requesting client; or if the PDC disallows the request, null ACI is returned.
- **BindObjectACI(objectID, objectACI)** request. This request allows the policy cognizant client to request that the PDC bind new object ACI to the specified object. It is expected that the PDC will require that the policy cognizant client possess an appropriate privilege.

The X Server is required to mark the **AccessAnswer()** and **BoundACI()** events with a tag indicating that the answer is genuine; otherwise a malicious client could use the generic X Window event mechanisms to spoof the PDC.

These mechanisms provide a simple method for a client to find out what policy the PDC is defining. For simple policies (e.g. strict MAC with a few labels), this method is sufficient; for more complex policies, it may be necessary for the policy defining client to make assumptions about the PDC policy. The problem is analogous to the problem of a client that wishes to be cognizant of the window management policy; in that case, the X Protocol also provides only basic information about the policy and a client needing to know more would have to make assumptions based on the documentation describing the window manager.

---

7. UNIX is a trademark of Unix System Laboratories, Inc.

2.4 Backward compatibility

It is important for an X Server implementing the above mechanisms to maintain a *backward compatibility mode* so that customers can choose to enable or disable the security policy as desired. The backward compatibility mode is simple in this case: if no client declares itself as a PDC, the X Server will not issue any **RequestAccess()** events, and the server will implement its original policy. Also, should a policy cognizant client issue a **RequestAccess()** or **GetACI()** request, the X Server will always return an **AccessAnswer()** *yes* or **BoundACI()** *null*, respectively. This retains interoperability in backward compatible mode.

2.5 Security through encapsulation or separation

Several vendors have attempted to implement security policy by *encapsulation* or *separation*, whereby the X Server runs untrusted. In an *encapsulation* architecture, there is typically a small, trusted X Server emulator which handles a limited set of trusted windowing operations [4]. Alternatively, for the *separation* architecture, a secure network and trusted X clients could be configured to implement secure windowing policy, without the need to implement a multilevel X Server.

These architectures can be made interoperable by defining a simple PDC that emulates the original X Window policy by giving a "yes" answer to any policy questions. (A PDC must be defined; otherwise, another client could spoof the PDC simply by doing an **AssertPrivilege()** request.) In general, encapsulation or separation architectures are defining virtual machines, where individual clients (even security-cognizant ones) *should not* be aware of the underlying window security policy.

2.6 Performance

The proposed protocol framework has performance implications. In particular, nearly every X Window request will cause the generation of one or more **RequestAccess()/AccessAnswer()** transactions. This potential performance problem can be solved or mitigated in several ways.

The first way is by taking advantage of local configurations. Typically, a machine supporting the X Window System also supports an in-memory local client connection facility, whereby clients running on the same machine as the X Server communicate via shared memory. If the X Server and the PDC are on the same machine, the **RequestAccess()** overhead should be considerably lessened.

The second way, which is an extension of the first, is for X terminals to support the proposed protocol[8]. Typically, X terminals provide enough memory for some clients to reside in the terminal firmware along with the X Server. X terminals also often provide downloading capabilities. Such capabilities could be used to create an X terminal-local configuration similar to the first method above. In this case, the performance should be even better, because the X terminal is dedicated to X Window operations.

The third way is for the X Server to provide an access decision cacheing facility, where the PDC's prior decisions are remembered by the server for later decisions. It is expected that many **RequestAccess()**s will be identical (or at least that the relevant ACI will be identical for many **RequestAccess()**s), so there would be a high cache hit ratio over the lifetime of an X Window invocation.

While it is possible to define a reasonable cacheing scheme to take advantage of these properties, it is not yet clear if it is really necessary. Such a scheme would complicate the protocol[9], and in order for the scheme to be interoperable, vendors would have to agree on a particular cacheing protocol, perhaps before the problem is completely understood. For these reasons, we have chosen not to include a cacheing scheme in the proposed framework.

---

8. The possibility of this happening is largely predicated upon the protocol being standardized by the MIT X Consortium as described earlier.

9. This complication would have to include, among other things, a cache flushing mechanism to allow the implementation of *time-based* policies such as RELEASEABLE AT <time>--see [3].

The fourth way is to implement the PDC not as a separate client but as a dynamically loadable library linked to the X Server. This method has been prototyped by the author, using library procedure calls in place of protocol transactions between the X Server and PDC, as a basic proof of concept of the protocol interface and as a simple performance modeling mechanism. The prototype implemented a very simple DAC policy. It did *not* contain the policy cognizance mechanisms. The prototype confirmed that many **RequestAccess()/AccessAnswer()** transactions occur in the startup phase of the X clients from the standard MIT distribution. However, after this small initial delay, no other delays based on this simple policy were noticeable. More performance modeling must be done with more complex policies before the mechanisms can be deemed practical.

3. Implications for X Server and Policy Defining Client

The proposed framework makes certain assumptions about the behavior of the X Server and the PDC.

First, the framework does not indicate what, if any, steps should be taken by the X Server or the PDC to alleviate denial of service attacks. For example, the framework does not dictate that only trusted clients be able to use the **XGrabServer()**[10] request. There are many other ways that a malicious client could degrade service through normal X Window requests. For the proposed framework, the X Server should be able to translate at least some of these denial of service problems into access requests that the PDC can act upon. For the example above, one reasonable solution would be for the X Server to define a SERVER object type and to issue an **RequestAccess()** requesting WRITE access to that SERVER object for the requesting client. The PDC can then decide if it should restrict access to this particular operation.

To generalize somewhat, the proposed framework assumes that the decomposition of the X Server into objects, object types, and access methods be done in such a way that *all reasonable policies* can be implemented. It is not clear by which criterion one should classify policies as reasonable; however, experience with existing secure X Window system models should be very helpful in this regard. One possible difficulty here is the creation of an X Server that is cognizant of the relationship between requests and events so that covert channels can be treated as access requests[11].

A further assumption is that the infrastructure (the secure OS and the secure network) provide facilities that do not compromise the security policies defined by the PDC. For example, there must be a trusted path to the PDC so that another client cannot spoof it. Also, the X Server must not allow access to an object when that object's ACI is being bound (the *tranquility* property). Finally, there must be a trusted path between the X Server and the physical display to preserve the integrity of any security relevant output (e.g. visible labels) or security relevant input (e.g. a security marking created at the user's discretion).

It is important to note that the implementation of the proposed framework alone is not sufficient for the X Window System to be certifiable past B1 or B1/CMW. A modular restructuring and covert channel analysis of the X Server, or possibly the implementation of an *encapsulation* or *separation* trusted X architecture as previously described, would also be necessary preconditions for B2 or B3 certifiability.

4. Implications for X Window System Embedded Base

A major advantage of a policy-free interface to the X Server is that the vendor can decide what impact the security policy will have on the embedded COTS (commercial off the shelf) client base. For example, the vendor may choose to implement a restricted form of the ss-property by making all objects invisible to a client unless their MAC labels are equal. Such a policy would tend to be useful in a system where the customer site is interested in strict separation of labeled data; however, such a policy has an impact on administrative COTS clients such as *xlswins* that

10. **XGrabServer()** tells the server to listen only to the client issuing the request until further notice.

11. Epstein [4] notes that the most difficult of the covert channels is the window exposure problem whereby one client can signal another client through the exposure of a previously covered window. The X Server must have an **RequestAccess()** strategically placed so that the PDC can determine the ACI of the exposing client and of the exposed window, and make a decision based on their relationship.

is different from the impact of a "read down" policy.

Often, the vendor will be choosing between defining a policy that reports that an access failure is due to the nonexistence of an object, and a policy that reports that the failure is due to a security violation. The vendor also has the opportunity to construct clever subpolicies such as (for example) defining certain objects as public, in order to provide greater compatibility with a particular COTS client.

The vendor can also make use of sophisticated policies in the attempt to provide compatibility. For example, a vendor willing to analyze the behavior of a particular COTS client might write a PDC defining a particular privilege that allows the client to access data that the invoking subject could not.

5. Implications for Other Trusted Systems

It has been noted [2] that the historical absence of a security policy has hampered the effort of reaching a consensus on a secure X Window system, largely because vendors have tailored the X protocol to provide the level of security and compatibility that they thought necessary. However, it is also true that the absence of a standard protocol has allowed vendors to explore many implementation possibilities, and in so doing there are now a set of de facto requirements for the support of various policies and implementations in any standard. From this point of view, the X Window system is in a superior position relative to other systems with premature de facto standard policy interfaces. Probably the best example of a premature policy standard is the UNIX discretionary access control policy and implementation, which (1) cannot be changed, and (2) cannot be described in fewer than ten complex rules(!).

It is much easier to *add* (re-engineer) a new policy interface than it is to *change* (reverse engineer) an existing one. The reverse engineering problem is that the goals of security (requiring a clear formulation) and compatibility (requiring no change to an old, unclear formulation) are at odds. This in turn implies that a rearchitecture of an existing system such as UNIX along the lines of the proposed framework would be problematic at best. The problem has been faced by the ORGCON prototype project [3] (among others); in that case, architectural purity was sacrificed for expediency.

For newer trusted systems, the methodology implied by the proposed framework has general applicability. Specifically, the trusted system designer is forced to face the following question: Given a target policy problem space, what is the simplest and best-performing mechanism that can be built that will support the entire policy problem space? It has often happened that the general applicability of a trusted system is not realized until it is fielded and new security requirements are generated based on field experience. The separation and decomposition methods described above would be a hedge against this eventuality.

6. Conclusions

By abstracting the security policy decision-making function away from the policy enforcement function, a simple, mechanistic interface will often become apparent. Such an interface has the potential of being both *non-controversial* and *extensible* to a large class of security policies.

We examined the effects of constructing such an interface for the X Window System, whose very reason for existence is to support a large, distributed, heterogeneous, open, and evolving graphical user interface environment. The approach appears to be most promising in systems with those qualities; the applicability of the approach is less evident in systems that are relatively small, monolithic, proprietary, or unchanging over time.

References

[1] Grenier, G., R. C. Holt, and M. Funkenhauser, Policy vs Mechanism in the Secure Tunis Operating System, *IEEE*, p. 84, 1989.

[2] Faden, G., Reconciling CMW Requirements with Those of X11 Applications, *Proceedings of the 14th National Computer Security Conference*, Washington, D.C., October 1-4, 1991.

[3] Abrams, M. et al, Generalized Framework for Access Control: Towards Prototyping the ORGCON Policy, *Proceedings of the 14th National Computer Security Conference*, Washington, D.C., October 1-4, 1991.

[4] Epstein, J. and J. Picciotto, Trusting X: Issues in Building Trusted X Window Systems or What's not Trusted About X?, *Proceedings of the 14th National Computer Security Conference*, Washington, D.C., October 1-4, 1991.

[5] Rosenthal, D., *LINX--a Less INsecure X server*, Sun Microsystems, April 1989.

[6] Picciotto, J., *Trusted X Window System*, MTP 288, The MITRE Corporation, February 1990.

[7] Scheifler, R. and J. Gettys, *X Window System*, Digital Press, 1990.

[8] Gettys, J., R. Scheifler, and R. Newman, *Xlib--The C Language X Interface*, Silicon Press, 1989.

[9] Graubart, R., J. Berger, and J. Woodward, *Compartmented Mode Workstation Evaluation Criteria, Version 1 (Final)*, DIA Directorate for Information Services, 1991.

[10] O'Brien, R. and C. Rogers, Developing Applications on LOCK, *Proceedings of the 14th National Computer Security Conference*, Washington, D.C., October 1-4, 1991.

[11] Carson, M. et al, Secure Window Systems for UNIX, *Proceedings of the 1989 Winter USENIX Technical Conference*, San Diego, CA, Jan 30-Feb 3, 1989.

[12] *Access Control Framework*, CD10181-3, ISO/IEC JTC 1/SC 21 N6188, June 24, 1991.

[13] Department of Defense, *Department of Defense Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, December 1985.

[14] *DNSIX 3.0 Architectural Overview, Rev 1*, SecureWare, April 1992.

USE OF A CASE TOOL TO DEFINE THE SPECIFICATIONS OF A TRUSTED GUARD

Robert Lazar
The MITRE Corporation
6301 Ivy Lane, Suite 304
Greenbelt, MD 20770
Contract No. NASW-4358

James H. Gray, III
Computer Sciences Corporation
10110 Aerospace Road
Lanham-Seabrook, MD 20706
Contract No. NAS-5-31500

Abstract

The authors of this paper were given the task of developing verification documentation that would serve the needs of system maintenance, but would also have the characteristic of being demonstrably correct. The result is a mathematically rigorous system specification using the Ward-Mellor variation of the Yourdon-DeMarco notation. The information content of this specification is virtually identical to that of an earlier system specification written in a formal specification language. In addition, the information content of this specification is computer program accessible, making feasible the development of an automated theorem prover.

Keywords: CASE Tools, Guard Processor, RAP, Restricted Access Processor, Structured Design, Trusted Software, Verification

INTRODUCTION TO THE RESTRICTED ACCESS PROCESSOR (RAP)

In December 1979 the National Aeronautics and Space Administration (NASA) and the Department of Defense (DoD) established the Network Security Ad Hoc Working Group (NSAHWG) to evaluate the impact of the DoD security requirements on the NASA Tracking and Data Relay Satellite System (TDRSS). One of the NSAHWG recommendations led to the development of the Restricted Access Processor (RAP) security guard at the Goddard Space Flight Center's (GSFC) Network Control Center (NCC). The NCC is a cleared facility that schedules and monitors the use of the TDRSS facilities, including support for Space Shuttle missions. The uncleared users must communicate with the NCC through the RAP in order to ensure that the unclassified message traffic strictly adheres to predetermined message formats and contents [4, 5]. The RAP received its Designated Approving Authority (DAA) accreditation in March 1986 and has operated as the automated NCC security guard since that date.

DESCRIPTION OF THE ORIGINAL VERIFICATION DOCUMENTATION

At the top of the RAP's original formal verification documentation stands the Air Force Security Policy for the RAP [8]. The essence of the Air Force Security Policy was translated into the RAP Formal Security Model [8], an abstract mathematical model written in the SYSPECIAL specification language [10, 15, 18]. A more concrete description of the RAP's security states and transition rules is found in the Top Level Specification (TLS), also written in SYSPECIAL [12, 16]. A semi-automated theorem proof checker was developed to prove that the TLS was consistent with the RAP Formal Security Model [13, 17]. The TLS views the security kernel of the RAP as a single black box. It models the RAP as a state machine. The SYSPECIAL specification language is built around the first order predicate calculus but has a syntax reminiscent of high order programming languages like PASCAL.

Between the RAP's original formal verification documentation and its corresponding software implementation a voluminous set of informal verification documents was developed. The Second

Level Specification (SLS) and Third Level Specification (3LS) documents (both in SYSPECIAL) described the RAP's security kernel at the processor and task levels, respectively. The TLS/SLS Correspondence and the SLS/3LS Correspondence documents painstakingly drew parallels between state transitions at one level with the more detailed state transitions at the next lower level. The task-level 3LS document was translated into English language Assertions. For each transition rule in the 3LS the Assertions described the initial conditions and the resultant final effects. The last verification step was accomplished by writing the English language Arguments, a series of task-specific documents which argued that the software implementation satisfied all of the desired properties as stated in the Assertions.

DESCRIPTION OF THE STREAMLINED VERIFICATION DOCUMENTATION

Because the original RAP verification documentation proved to be too difficult to maintain due to its sheer volume as well as its highly abstruse nature, the RAP project manager at GSFC decided to pursue the development of a streamlined and simplified verification methodology.

As the new verification documentation was to also serve as the main documentation for the maintenance of the RAP system, it also had to have the characteristics of good system documentation. Whereas the original verification documentation was voluminous and abstruse, the new verification documentation was to be concise and readable. Whereas the original documentation was difficult and costly to change, the new documentation would be easy and inexpensive to change. In addition, the new documentation would have to be demonstrably correct. Thus two traditionally separate kinds of documentation would be combined into one. It was expected that tremendous savings in time and costs would be realized for the whole process of maintaining the RAP system.

A committee of RAP contractors was formed called the RAP Methodology Working Group (RMWG). The RMWG realized that the security requirements for the RAP had been formulated prior to the publication of the DoD Trusted Computer System Evaluation Criteria (i.e., the "Orange Book," [6]) and that the RAP verification documentation could not easily be retrofitted into a strict conformance with any Orange Book security level. Nevertheless, the RMWG felt that it was valuable to follow the spirit of the Orange Book as closely as was feasible [2, 3].

The Computer Security Center document CSC-STD-004-85 (commonly called the "Yellow Book," [7]) suggests the B2 level as a minimal documentation level for the RAP since the RAP is a "closed system" which protects data rated as high as SECRET from users who may be uncleared (Table 7 p. 21 of [7]). However, in the "Design Specification and Verification" sections of the Orange Book, the only difference between the B2 and B3 descriptions is the following additional B3 statement: "A convincing argument shall be given that the DTLS [Descriptive Top Level Specification] is consistent with the model." The RMWG decided to emulate the B3 verification documentation level. The B3 Design Specification and Verification requirements as given by the Orange Book may be summarized as follows:

1. There shall be a formal model of the security policy which is "proven consistent with its axioms."

2. There shall be a DTLS which is "shown to be an accurate description of the TCB [Trusted Computing Base] interface."

3. There shall be a "convincing argument" which demonstrates that the DTLS is consistent with the formal model.

Requirement 1 is met by the fact that there is a mechanism for proving the TLS consistent with the much smaller RAP Formal Security Model. The RMWG decided to change nothing with respect to how the RAP satisfied requirement 1. However, for requirements 2 and 3, the RMWG decided to utilize the CASE tool technology to develop a DTLS for the RAP. Requirement 2 would then be satisfied by showing that each object in the DTLS mapped to an implementation in the code; similarly, each code segment would be required to map to an object in the DTLS. A similar correspondence would need to be developed between the formal model and the DTLS, thereby providing the Convincing Argument demanded by requirement 3.

An overview of the new verification documentation and its derivation from the old is shown in Figure 1. Certain elements of the old documentation are carried over unchanged into the new. These include the Security Policy, the PDL, and the code. The new Formal Security Model is constructed from the old Formal Security Model and the old TLS (see [19] for a discussion of the abstract and concrete parts of the Formal Security Model). The DTLS is completely new. The elements of the old documentation shown in the dotted box (SLS, 3LS, Assertions, and Arguments) are being discarded because those documents are the most voluminous and have to be maintained manually. The DTLS, Convincing Argument, and Traceability Tables provide the new bridge between the Formal Security Model and the code [1, 9, 14].

DEVELOPMENT OF THE DESCRIPTIVE TOP LEVEL SPECIFICATION

The DTLS is the highest level document outlining the implementation of the design imperatives set forth in the concrete part of the Formal Security Model. Its language is the Yourdon-DeMarco notation of Data Flow Diagrams, State Transition Diagrams (STDs), Decision Tables, and structured English [11, 20]. It is the primary design document for the RAP. The DTLS follows rigorously from the concrete part of the model, but unlike the SYSPECIAL specification language of the model, it is readily usable by any programmer or analyst schooled in the Yourdon-DeMarco methodology.

The Ward-Mellor variation of the Yourdon-DeMarco notation can be as mathematically exact as any formal specification language. The Data Flow Diagrams unequivocally define the genesis, transformation, and disposition of every piece of data. The STDs and Decision Tables can give an unambiguous definition of any state machine. The data dictionary ensures that every entity is defined, and the CASE tool enforces the data dictionary definitions.

Key to the success of this project is being able to show by "Convincing Argument" that the DTLS is consistent with the Formal Security Model and also to show by means of the Traceability Tables that the PDL and code are consistent with the DTLS.

The SYSPECIAL specification language is built around three kinds of entities: VFUNS; OFUNS; and Function Definitions.

The VFUNS are value functions (i.e., data structures) which collectively define the state space of the RAP at any given point in time. All security-relevant actions of the RAP either transmit data across the security boundary or change a value of one of the VFUNs.

The OFUNs specify the operational functions by which the RAP achieves its objectives. They define the concrete part of the Formal Security Model. At the beginning of each operation, the RAP is at a certain point in the state space. Local assertions in the OFUNS determine whether or not, given the RAP's position in the state space, the operation can proceed. The effect of the operation is to change the RAP's position in the state space.

The Function Definitions provide the semantics of the Formal Security Policy Model. Some of the Function Definitions are predicates, mapping from statements in SYSPECIAL to the values

TRUE and FALSE. Other Function Definitions map from points in the RAP state space to other points in the RAP state space.

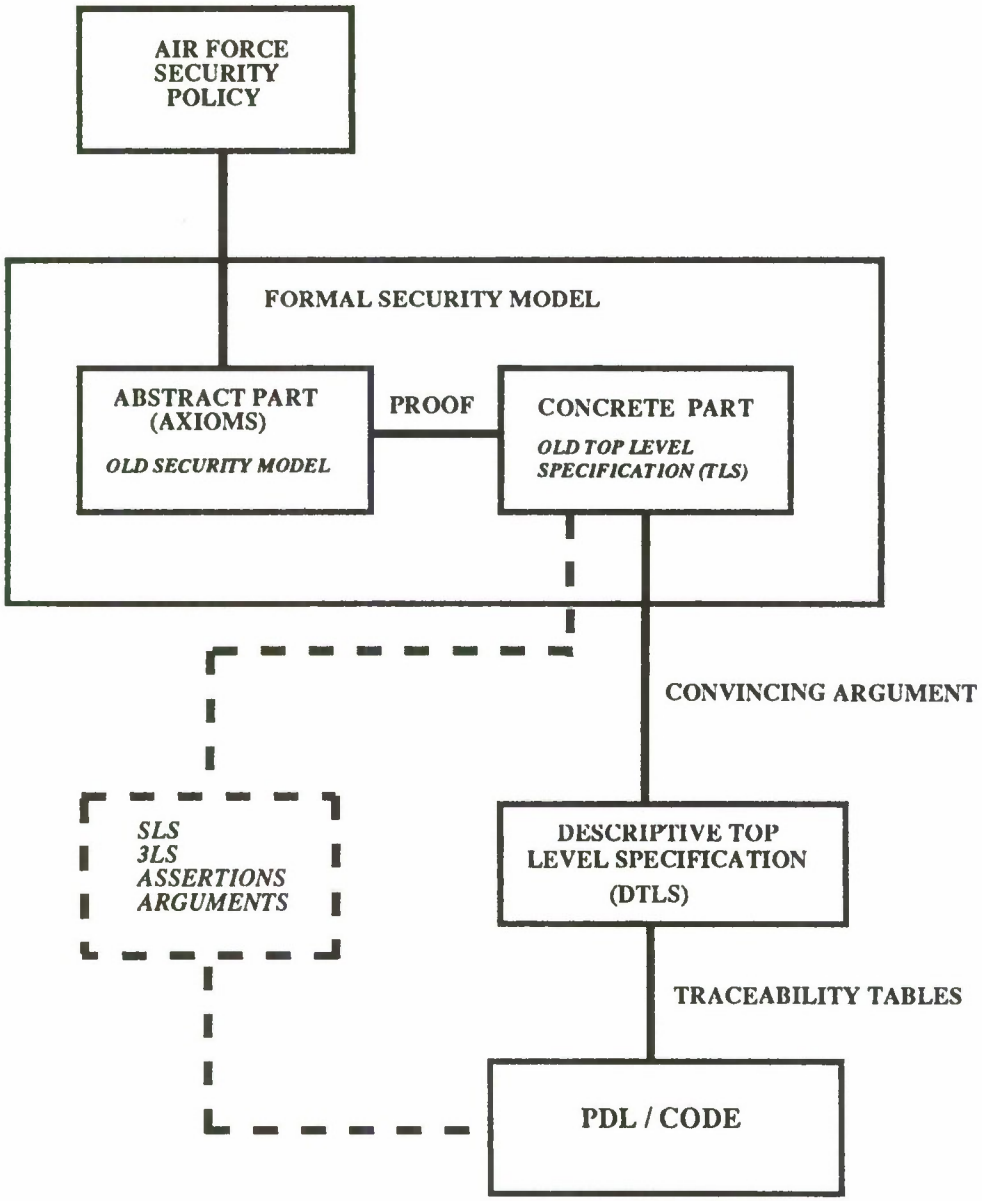


Figure 1. Overview of the New RAP Documentation and Verification Methodology

Note: Elements of the old methodology that are being dropped are shown in the dotted box.

The DTLS was developed one functional area at a time. Every effort was made to preserve intact the meanings of the SYSPECIAL specification statements. The first step was to sort the OFUNs of the TLS into groups of operations that for each group accomplished one global function. For each functional area, a first cut at the DTLS diagrams was made based on the TLS and the SLS. The corresponding software was then consulted to confirm, correct, or refine the

diagrams. Almost always, the inspection of the code uncovered misinterpretations of the TLS or SLS which had to be corrected. The development of the DTLS was thus an iterative process. The CASE tool was invaluable in keeping the set of hierarchical diagrams balanced. For functional areas not yet developed, a stub was introduced as a placeholder.

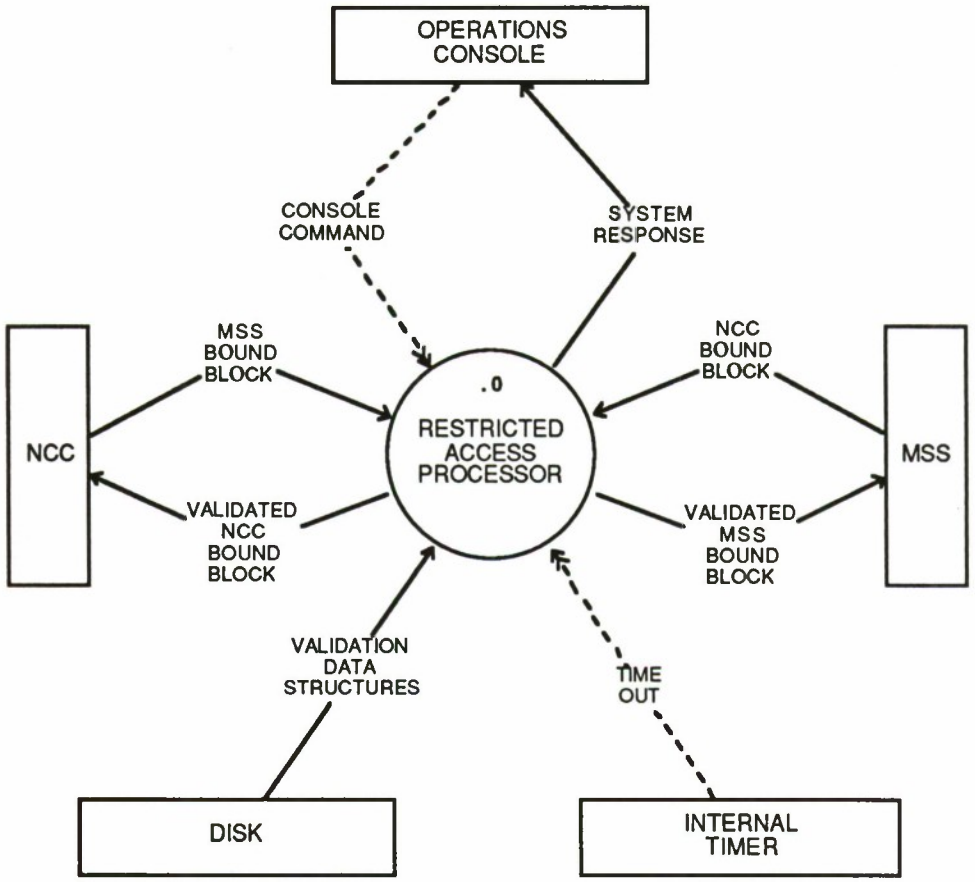


Figure 2. RAP DTLS Context Diagram

All of the Data Flow Diagrams in the DTLS were mapped directly from the TLS and certain portions of the SLS. Simplified examples of the Data Flow Diagrams in the DTLS are given in Figures 2 and 3.

The context diagram is given in Figure 2. The single process bubble represents the RAP and the five terminators are represented by labeled rectangles. These five terminators are mapped directly from the five ports defined in the TLS: NCC (Network Control Center), MSS (Message Switching System), UIS (User Interface System or Operations Console), DISK (Validation Data Structures), and INTERNAL (Interval Timer). The various data flows and control flows represent: the various message blocks passing through the RAP; the downloading of the Validation Data Structures; the commands from the operations console; the system responses to those commands; and the timeout signals from the internal port.

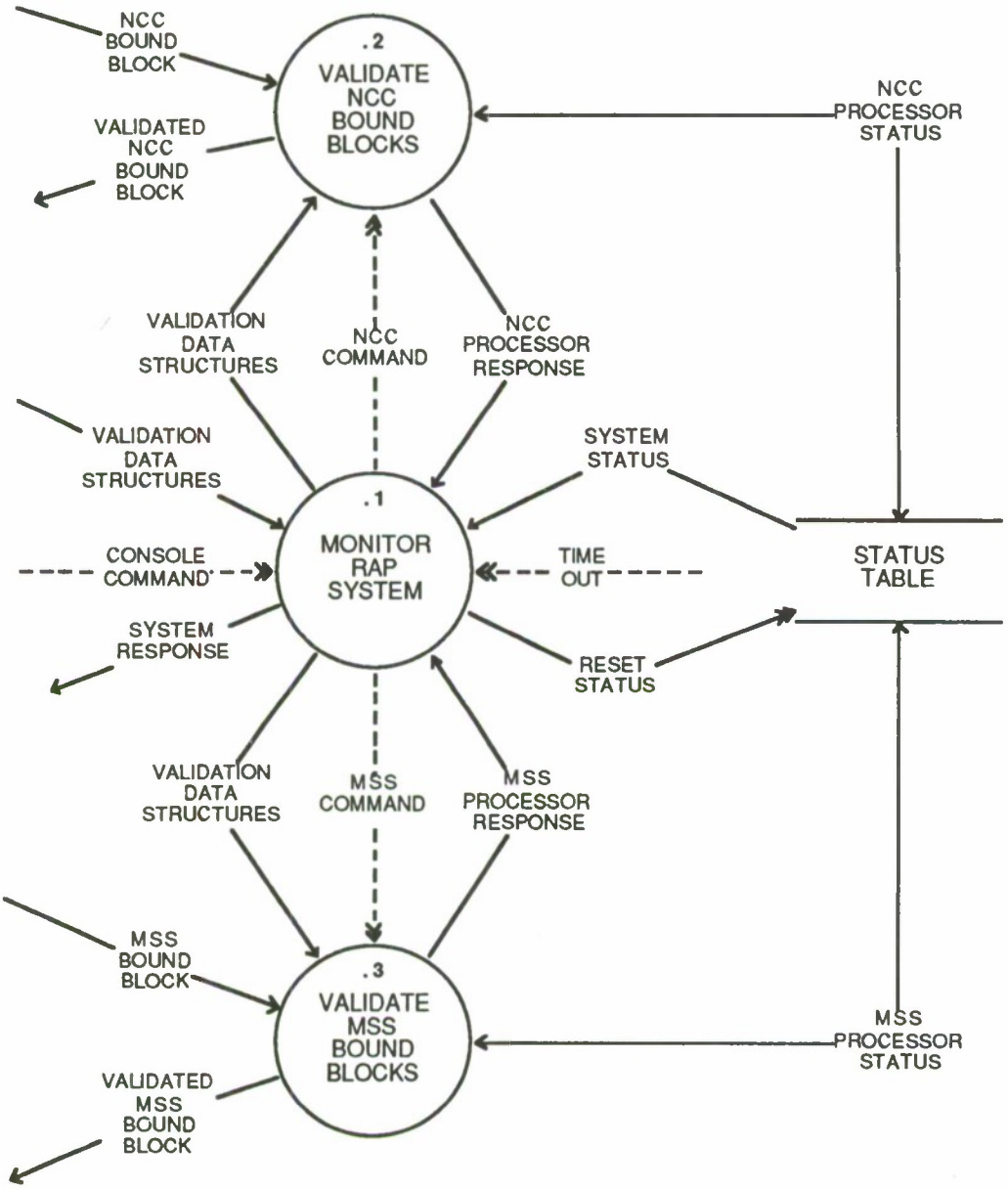


Figure 3. RAP DTLS Level 0 Data Flow Diagram: RESTRICTED ACCESS PROCESSOR

The level 0 Data Flow Diagram is given in Figure 3. This highly simplified diagram describes the operation of the RAP system. Message blocks passing through the RAP are validated by one of the two validation processors. Only those found to be valid are passed on. Those found to be invalid are not. Invalid message blocks are also logged, but for purposes of simplification, this operation is not being shown in these illustrative diagrams. The Validation Data Structures describe the valid formats for the various types of message blocks passing through the RAP. The operational status of the processors is kept in the Status Table and updated at periodic intervals. If any of the processors does not update the Status Table before the timeout signal for that interval, the whole RAP system is shut down. This is in keeping with the Air Force Security Policy which states that the RAP may not process messages in degraded mode. Each of the three process

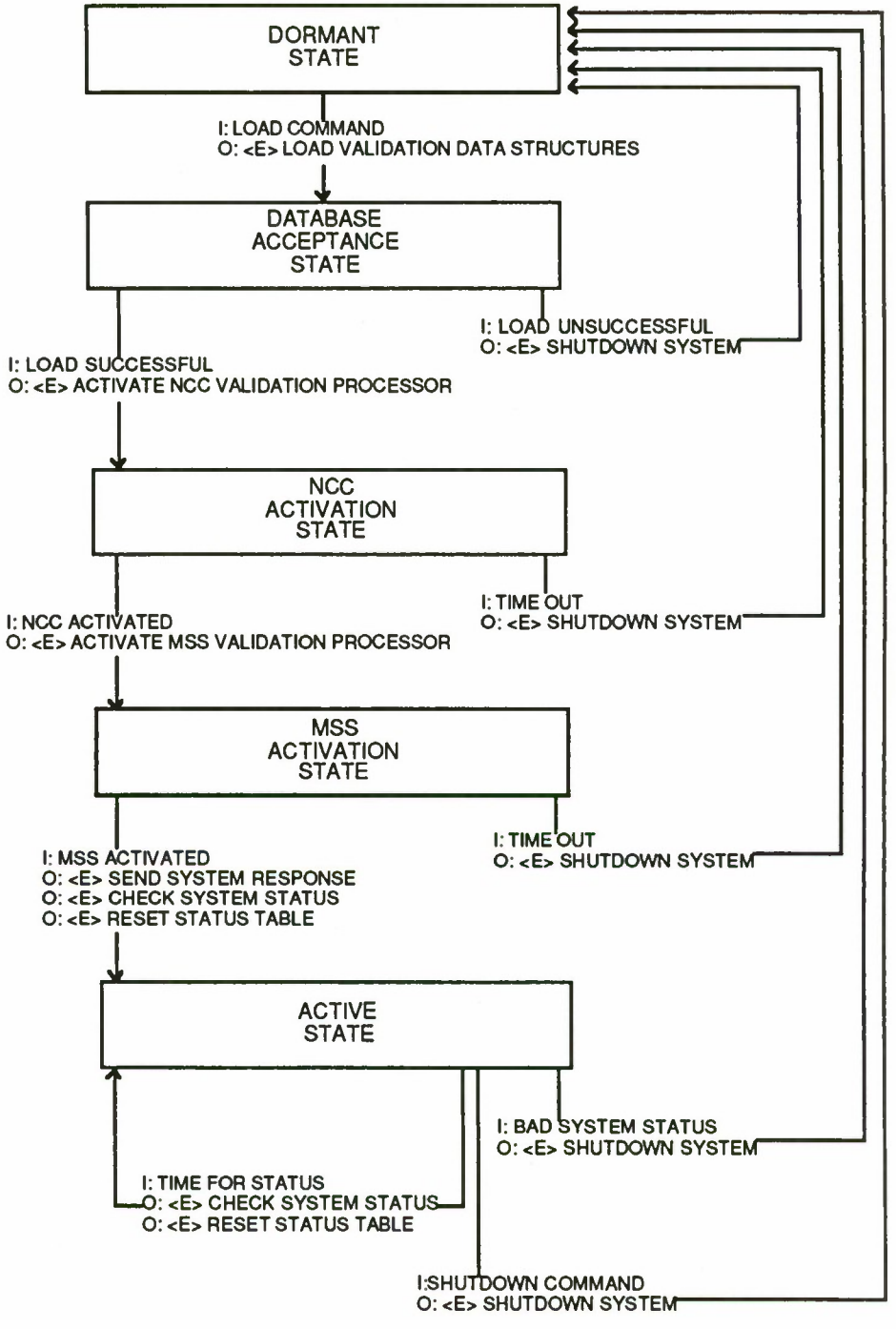


Figure 4. RAP DTLS State Transition Diagram: RAP EXECUTIVE

bubbles represents one of the major functional processor groups. The solid lines represent data flows between the processor groups while the dotted lines represent flows of control. The parallel horizontal lines represent the Status Table.

Figure 4 gives an example of an STD. This highly simplified diagram is mapped from the operational functions (OFUNS) which define the processing of the Operations Console commands for downloading the Validation Data Structures and for otherwise controlling the block validation processors of the RAP system. As the diagram shows, if any of these operations fail or do not take place before the timeout signal, the system is shut down. The system is also shut down if any processor's status report does not arrive before the timeout signal for that status reporting period.

The derivation of the STDs follows naturally from the state machine nature of the Formal Security Model. Each of the operational functions in the model (the OFUNS) defines a state transition. The assertions in the operational functions define those conditions which must be true and those events which must occur in order for the state machine to move from one given state to another. The effects in the operational functions define the state that the machine will transition to as well as the actions taken by the machine. These actions include outputs across the security perimeter, the assigning of values to various global data structures (VFUNS), and the initiation of data flows between the various processors making up the RAP.

Each large box in the STD represents a state. The arrows, called "transition arcs," show the transitions from state to state. Each transition arc is labeled by the events which trigger the transition and the accompanying actions taken by the RAP. The events which trigger the transition are displayed as input items (prefaced by "I:" on the figure). The actions taken by the RAP, concurrently with the change in state, are displayed as output items (prefaced by "O:" on the figure). Often an output item activates (or enables) a process on a Data Flow Diagram, as is indicated on the figure by "<E>" followed by the process name.

DEVELOPMENT OF THE CONVINCING ARGUMENT

The Convincing Argument is an intuitive demonstration that the DTLS follows from the concrete part of the Formal Security Model (i.e., the TLS). In essence, it is a table of correspondences between the elements of the concrete part of the model and the elements of the DTLS. It seeks to make explicit the mapping from the one-dimensional PASCAL-like notation of the SYSPECIAL specification language to the two-dimensional pictographic notation of the Yourdon-DeMarco diagrams. For example, in relating the STD of Figure 4 to the OFUNs from which it is derived, the Convincing Argument presents explicitly the correspondences between the assertions and effects of those OFUNs and the transition arcs of Figure 4. Similarly, the states of Figure 4 and data stores such as STATUS TABLE in Figure 3 are related back to their antecedent VFUNs. Basically, the Convincing Argument is a proof by inspection. It directs the reader to each OFUN, VFUN, and Function Definition of the Formal Model and invites the reader to decide for him- or herself whether the corresponding elements of the DTLS maintain the security-preserving state transitions of the model.

NOTE: The SLS was consulted during the development of the DTLS in order to get a good first cut at the nature of the interprocessor communications which are not specified in the TLS black-box specification.

DEVELOPMENT OF THE DTLS TRACEABILITY TO THE RAP CODE

As the DTLS was produced for the various RAP functional areas, the code was consulted to confirm or correct the first cut at the diagrams as obtained from the TLS and SLS. All of the Data Flow Diagrams were eventually defined in terms of leaves of the hierarchical tree, namely, in terms of structured English Minispecs, STDs, or Decision Tables. A complete traceability of the DTLS to the code is achieved by making correspondences between all of the DTLS leaves and all of the code implementing each leaf. Typically a Minispec corresponds to one contiguous segment of code which manipulates a set of inputs to produce a set of outputs, whereas separate decision paths on an STD correspond to separate sections of code. Decision Tables can be similar to Minispecs if

they merely produce a set of output variables based upon the values received for various input variables. On the other hand, a Decision Table becomes a control process (similar to an STD) if it uses the input variables to decide which processes to activate next. A Decision Table can simultaneously produce output variables and enable the execution of other processes. All of the threads of execution in the code should have corresponding sequences of actions in the DTLS in order for the traceability to demonstrate that the code conforms to its verification specifications.

CONCLUSIONS AND POSSIBLE FUTURE DIRECTIONS

This project demonstrates that a system specification written in a formal specification language may be functionally duplicated with a CASE tool using the Ward-Mellor [20] variation of the Yourdon-DeMarco notation. The TurboCASE (see NOTE at bottom of page) tool with real-time extensions is a natural for translating the TLS finite state machine representation into a Ward-Mellor structured representation. The entities of the Formal Security Model map easily and naturally into the entities of the DTLS. The Formal Security Model is rigorous and mathematically exact. The DTLS is also rigorous and exact in its own right, but is written in an intuitive notation that is easily learned and by now familiar to most software engineers.

The TurboCASE (see NOTE at bottom of page) tool automatically performs checks for consistency and balance. Previously, most tasks, such as establishing the correspondences between the TLS/SLS/3LS levels of detail, had been done manually by the security analyst.

If it is desirable to eventually phase out all use of the SYSPECIAL specification language in the RAP verification documentation, several options exist. If the B1 security level is considered to provide sufficient assurance for a trusted guard such as the RAP, then the DTLS is a strong candidate for the B1 informal model. On the other hand, if the assurance level needs to be maintained at the B2 or B3 security level, then a proof mechanism is necessary that does not rely upon the SYSPECIAL language. The abstract and concrete parts of the Formal Security Model are both written in SYSPECIAL and the concrete part (the TLS) has been successfully translated into the Yourdon-DeMarco notation. The abstract part may be similarly translated using the CASE tool. An ASCII coded data interchange file provided by the CASE tool makes all of this specification information accessible to any computer program. This makes feasible an automated theorem prover for the DTLS.

REFERENCES

[1] Belton, M., 1990, *Iconix and TurboCASE CASE Tool Evaluation Report for the Restricted Access Processor (RAP)*, CSC Contract NAS-5-31500, NASA, Goddard Space Flight Center, Greenbelt, Maryland. (see NOTE at bottom of page)

[2] Booz-Allen & Hamilton Inc., 1989, *RAP Transition Plan,* NAS-5-30169, NASA, Goddard Space Flight Center, Greenbelt, Maryland.

[3] _____, 1990, *RAP Overview Document,* NAS-5-30169, NASA, Goddard Space Flight Center, Greenbelt, Maryland.

[4] Computer Sciences Corporation, 1983, *Restricted Access Processor (RAP) Computer Program Development Specification (B5): Block Validation Program (BVP)*, CDRL 110-1, NASA, Goddard Space Flight Center, Greenbelt, Maryland, section 3.2.

NOTE: TurboCASE is a registered trademark of StructSoft, Inc.;
PowerTools is a registered trademark of Iconix Software Engineering, Inc.

[5] _____, 1983, *Restricted Access Processor (RAP) Computer Program Development Specification (B5): Gateway Program (GWP)*, CDRL 110-2, NASA, Goddard Space Flight Center, Greenbelt, Maryland, section 3.2.

[6] Computer Security Center, 1985, *Department of Defense Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD, DOD, Computer Security Center, Fort George G Meade, Maryland.

[7] _____, 1985, *Technical Rationale Behind CSC-STD-003-85: Computer Security Requirements. Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria*, CSC-STD-004-85, Computer Security Center, Fort George G Meade, Maryland.

[8] DiVito, B., 1985, *Restricted Access Processor (RAP) Formal Security Model*, CDRL 140-7, NASA, Goddard Space Flight Center, Greenbelt, Maryland.

[9] Gray, J. H., 1989, *Iconix PowerTools Evaluation Report for the Restricted Access Processor (RAP)*, CSC Contract NAS-5-31500, NASA, Goddard Space Flight Center, Greenbelt, Maryland. (see NOTE at bottom of previous page)

[10] Halpern, J. D., S. Owre, N. Proctor, and W. F. Wilson, February 1987, "Muse - A Computer Assisted Verification System," *IEEE Transactions on Software Engineering*, Vol. SE-13, No. 2, pp.151-156.

[11] Hatley, D. and I. Pirbhai, 1987, *Strategies for Real-Time System Specification*, New York, Dorset House Publishing Co.

[12] Lazar, R., 1989, *Restricted Access Processor: Annotated Augmented Top Level Specification*, MITRE contract, NASA, Goddard Space Flight Center, Greenbelt, Maryland.

[13] Lazar, R., 1990, *The RAP Proof: Exposition and Critique of a Formal Verification Methodology*, Report No. MTR 90W00021, MITRE Corporation, McLean, Virginia.

[14] Lazar, R., 1991, *The New RAP Methodology*, Report No. WP 91W00182, MITRE Corporation, McLean, Virginia.

[15] Owre, S., and J. D. Halpern, 22 July 1985, *MUSE: The Sytek Proof Processing System*, Report No. TR-85007, SYTEK Inc., Mountain View, CA.

[16] Proctor, N., 26 July 1985, *Restricted Access Processor (RAP) Top-Level Specification*, CDRL 110-5, NASA, Goddard Space Flight Center, Greenbelt, Maryland.

[17] _____, 26 July 1985, *Restricted Access Processor (RAP) Verification Results Report*, CDRL 141-3, NASA, Goddard Space Flight Center, Greenbelt, Maryland.

[18] Silverberg, B.A., L. Robinson, and K. N. Levitt, June 1979, *The HDM Handbook - Volume II: The Languages and Tools of HDM*, SRI Project 4828, SRI International, Menlo Park, CA.

[19] Tavilla, D. A., 22 June 1986, *A Guide to Understanding the Orange Book Security Model Requirements*, Report No. WP 26782, MITRE Corporation, Bedford, Mass.

[20] Ward, P. and S. Mellor, 1985, *Structured Development for Real-Time Systems, Volumes I, II, and III*, New York, Yourdon Press.

Tutorial Series On Trusted Systems

Presented by:

R. Kenneth Bauer
Joel Sachs
Dr. Gary Smith
Dr. William Wilson
Arca Systems, Inc.
2841 Junction Ave., Suite 201
San Jose, CA 95134
408-434-6633

Dr. Charles Abzug
LtCdr Alan Liddle, Royal Navy
Howard Looney
Information Resources Management College
National Defense University
Fort Lesley J. McNair
Washington, D.C. 20319
202-287-9321

Schedule

	Tuesday - 13 Oct 1992	**Wednesday - 14 Oct 1992**	**Thursday - 15 Oct 1992**
0900 -1030	(Conference Plenary)	Trust Fundamentals *Dr. Charles Abzug*	Trust Fundamentals *Dr. Charles Abzug*
1100 - 1230	(Conference Plenary)	Trusted Networks *R. Kenneth Bauer*	Trusted Networks *Joel Sachs*
1400 - 1530	Threats & Security Overview *LtCdr Alan Liddle*	Trusted Databases *Dr. Gary Smith*	Trusted Databases *Dr. Gary Smith*
1600 - 1730	Physical, Personnel, & Administrative Security *Howard Looney*	Trusted Integration & System Certification *Joel Sachs*	Trusted Integration & System Certification *Dr. William Wilson*

Description

These tutorials are based on courses and seminars given by either Arca Systems or the Information Resources Management College of the National Defense University. Arca's Information Security Seminars focus on several topics, including Computer, Network, and Database Security and incorporate experience from applying Arca's security consulting and engineering services on MLS systems solution developments. The Information Resources Management College includes security in its information management courses, particularly through an intensive Automated Information Systems Security Course which is taught in the graduate level.

The tutorials will be presented in lecture format with question and answer periods. While there is a logical flow between the tutorials, each tutorial will be presented as a separate unit so that conference attendees can attend any or all of them. The tutorials are intended to introduce many and varied security topics as opposed to exploring them in-depth. Brief descriptions of each tutorial identified above follows:

Threats & Security Overview focuses on the general threats to automated information systems, assets requiring protection, and an overview of security disciplines (operational, communications, computer, physical, and administrative) as well as their interrelation.

Physical, Personnel, & Administrative Security focuses on the concept of "layering" security through physical, personnel, and administrative security measures. Topics include: criteria for facility selection, design, and access controls, as well as environmental, fire protection, and power considerations.

Trusted Networks focuses on basic points in network security and gives an overview of the TNI. Topics include network security concerns and services, the NT and its Evaluation Classes, system composition and interconnection, cascading, and an overview of the NT evaluation process.

Trusted Database Systems focuses on security from a "database view" and gives an overview of the TDI. Topics include DBMS specific security requirements, vulnerabilities, and challenges; database design considerations; implementation issues; and use issues.

Trusted Integration & System Certification focuses on issues in integrating MLS solutions using trusted products, the development of the certification evidence, and the accreditation process. Topics include system-wide security and assurance, security trade-offs, and development methodologies.

Executive Summary

Addressing U. S. Government Security Requirements for OSI

PANEL SESSION OVERVIEW

Noel A. Nazario, Chair
Protocol Security Group
National Institute of Standards and Technology

Are U. S. Government requirements for protecting unclassified network data being met by emerging voluntary international standard protocols? This session focuses on security requirements that need to be addressed within the U. S. Government Open Systems Interconnection Profile (FIPS Pub 146-1) and current efforts to meet them. The session examines Government requirements, discusses emerging security protocols, and describes a labeling infrastructure necessary to support security services and protocols.

Federal agencies are experiencing a growing need to safeguard information exchanges on computer networks. While their security needs grow, communications systems currently available to agencies do not provide the necessary protection. Recently, federal agencies were mandated to transition their computer communications systems to the Open Systems Interconnection (OSI) architecture (IS 7498). OSI is a non-proprietary communications architecture developed by the International Standards Organization (ISO). At the present time, OSI does not provide adequate security features either. The National Institute of Standards and Technology is working to stimulate the development of OSI-based products that meet Government requirements, including security. NIST has produced a Federal Information Processing Standard (FIPS) 146-1 known as the Government Open System Interconnection Profile (GOSIP). GOSIP is a mandatory procurement specification for Government communications systems based on the OSI architecture. FIPS 146-1, GOSIP version 2, contains a single security option for the Connectionless Network Protocol (CLNP) and an appendix discussing standardization efforts in the area of security. GOSIP version 3, intended for the FY93 time frame, contains an updated security appendix, and references to a common profile called Industry and Government Open Systems Specification (IGOSS). This common profile is being developed jointly by NIST, the Manufacturing Automation Protocol (MAP) and the Technical and Office Protocol (TOP) communities, and the Electric Power Research Institute (EPRI). Its purpose is to unify the procurement and operational requirements of major U. S. users of OSI products. The IGOSS incorporates security features for the 1988 X.400 Message Handling System and X.500 Directory Services. Still, the X.400 and X.500 security features in the IGOSS fall short of meeting agency security requirements.

GOSIP is based on international standards and agreements reached by vendors of communications equipment participating in the Open Systems Environments (OSE) Implementors Workshop (OIW). The GOSIP outlines a subset of OSI services and features, required by the Government, that vendors have agreed to implement. The functionality outlined in GOSIP must reflect all agency requirements but, so far, security requirements have not been fully addressed.

In July 1988, the International Standards Organization (ISO) approved the OSI Security Architecture (IS 7498/2). It provides a framework for the incorporation of security to OSI protocols. Five primary security services, authentication, access control, data confidentiality, data integrity, and non-repudiation, are specified in the architecture. IS 7498/2 also discusses mechanisms that may be used to provide these services and the OSI layers where they could be offered. While the Security Architecture lays the ground work, significant effort is required to standardize protocol specifications that contain security features.

The five primary security services defined in IS 7498/2 provide safeguards against unauthorized access to systems and data, and against unauthorized disclosure, modification or destruction of data, which may occur accidentally or intentionally. The security services are described bellow.

- **Data confidentiality** services protect against unauthorized disclosure. Protection of medical records to insure patient's privacy is an example of the need for confidentiality.

- **Data integrity** services protect against unauthorized modification, insertion and deletion. Electronic funds transfer between banks requires protection against modification of the information.

- **Authentication** services verify the identity of communicating peer entities and the source of data. Owners of bank accounts require assurance that money will be withdrawn only by them.

- **Access control** services allow only authorized communications and access to system resources. Only financial officers are authorized access to a company's financial plans.

- **Non-repudiation**, with proof of origin, provides to the recipient, proof of the origin of data and protects against any attempt by the originator to deny sending the data. Non-repudiation, with proof of delivery, provides to the sender proof of the delivery of data. The non-repudiation service may be used to prove to a judge that a person received or sent a message (e.g., a purchase order).

Government agencies may require the implementation of all or some of these services in their communications systems. The Security Architecture indicates that authentication, confidentiality, integrity, and access control services may be implemented in layers three, four and seven of the OSI architecture. The non-repudiation service may only be offered at layer seven. The selection and placement of mechanisms to support security services is based on the perceived threats and a balance between protection and cost.

NIST is responsible, under the 1987 Computer Security Act, for developing standards for the protection of unclassified but sensitive information in Federal computer systems. In meeting that responsibility NIST coordinates efforts with other Government agencies and supports the national and international standards process. Based on perceived Government needs, NIST has identified areas where security enhancements to GOSIP are required. Figure 1 depicts planned security enhancements.

		Initial Enhancements	Future Enhancements
Protocols	Layer 7	• Private Key-Based Key Management Protocol (KMP)	• Public Key-Based KMP • Trusted Network Management • Secure Message Handling System • Secure Directory Services
	Layer 4	• Transport Layer Security Protocol (TLSP/SP4)	
	Layer 3		• Network Layer Security Protocol (NLSP/SP3)
Supporting Security Infrastructure		• Security Labels • Computer Security Objects Register	• Certificate Administration

Figure 1 - GOSIP Security Enhancements

These enhancements include both the establishment of a security infrastructure and the incorporation of security protocols. For instance, the initial enhancements include the adoption of a standard security labeling scheme and the establishment of a Computer Security Objects Register (CSOR). The standard labels provide for a uniform way of conveying security-related information about individual data units. The CSOR allows the registration of specific semantic definitions for generic protocol elements and the assignment of unique identifiers used to establish security parameters for communication. These initial security enhancements include a private key-based (symmetric) key management protocol based on ANSI X9.17 at the Application Layer and a security protocol for the Transport Layer.

Future enhancements will include the establishment of mechanisms for the administration of certificates for use with public key-based (asymmetric) cryptography. Certificates are unforgeable pieces of information that identify communicating entities. Certificates can be used to support access control, authentication, and non-repudiation services. Also planned is the incorporation of a security protocol for the Network Layer. Enhancements to the Application Layer include, secure messaging services, secure directory services, trusted network management, and a public key-based key management protocol.

Before security enhancements can be made to GOSIP, standards on which to base implementation agreements need to exist. OSI security standards being currently developed by ISO will be the basis implementors agreements by the OSE Implementors Workshop (OIW). Several sub-committees (SC21, SC27, and SC6) of the ISO/IEC Joint Technical Committee One (JTC1) is responsible for OSI standards on computer security-related matters. An overview of these bodies and the status of their activities will be presented by Mr. Ted Humphreys from XISEC

Consultants, Ltd. in the United Kingdom. Mr. Humphreys is an active participant in the international standards process.

The OSE Implementors Workshop convenes four times a year at NIST and produces agreements for the implementation of commercially available OSI products. Mr. Dale Walters, from the Systems and Network Architecture Division at NIST, will present the status of security-related work by various Special Interest Groups (SIGs) within the OIW. Mr. Walters participates actively in the Security and the Lower Layers Special Interest Groups.

NIST coordinates its security efforts with other government agencies and military organizations with their own security requirements and expertise. An instance of this cooperation is the development of a uniform security labeling strategy for GOSIP. Security experts from various Government and private organizations participated in this effort. Representatives from various organizations attended two workshops on security labels held by NIST and provided technical comments on the documents generated. These workshops helped NIST develop a security label specification defined in a proposed Federal Information Processing Standard (FIPS) called "Standard Security Label for the Government Open Systems Interconnection Profile." Professor Thomas Bartee, from the Institute for Defense Analysis, an original contributor to this work, who served as liaison between NIST and other government organizations with special security requirements discusses current issues in security labeling.

Another example of inter-agency cooperation is the establishment of registration procedures for computer security objects. This effort also originated from the discussions at the NIST security labeling workshops. From those discussions it became apparent that a separation between policy and implementation issues was necessary for the success of a generic security labeling strategy. The use of a Computer Security Objects Register (CSOR) provides for the specification of policy-driven handling and interpretation rules independently form the label specification. NIST is currently establishing such a register and coordinating registration procedures with the Defense Information Systems Agency (DISA).

The CSOR will not be limited to the registration of security labeling semantics. By assigning unique names to other types of security objects, it will assist in the negotiation of parameters for secure communications. All the emerging security protocols require that the communicating parties share prior knowledge of security services and mechanisms protecting the communication. Cryptographic keys for the provision of confidentiality services via symmetric key algorithms are examples of information that need to be shared prior to a secure data exchange.

As OSI security standards become available, future versions of GOSIP will incorporate emerging commercial offerings that meet Government requirements. NIST is placing special attention on security infrastructure issues such as security labeling, security object registration and the availability of appropriate key management systems for loading security parameters. These elements are central to the availability of all other security protocols and services.

Executive Summary

OSE Implementor's Agreements

Dale Walters
National Institute of Standards and Technology
July 27, 1992

This executive summary reflects on what security agreements have been reached in the OSE Implementors Workshop (OIW) as well as what still needs to be done in the next few years.

OIW

The OSE Implementor's workshop is tasked with taking base standards and producing working agreements that vendors will agree to build and users will procure. Security implementation agreements are finally in the process of being developed. This talk will give a brief overview of what Special Interest Groups (SIGs) are doing in the way of developing these working agreements. The areas that will be covered are lower layers (Transport and Network), Directory, and the overall architecture developed by the Security SIG.

Lower Layer

Initial working agreements have been started on both the Network and Transport Layer Security protocol. These initial agreements will form the basis for future GOSIP functional requirements.

Some of the agreements are as follows:

A) Ordering of security functions

B) Size of fields

C) Access control and integrity

Lower Layers has been tasked to develop encapsulation formats for use by both of the above mentioned standards. The profile, based on specific algorithms for integrity and confidentiality will indicate what the protocol data unit will look like and what will need to be encapsulated before sending the data from one system to another.

An important part of both the NLSP and TLSP standards is the Security Association - Protocol SA-P. An association between two entities must first be formed before secure communications can be started. This can be done at the application level, by manual exchange of information and also by the specific transport or network entity. The last example is well on the way to becoming an international standard and will be adopted by the OIW. I would like to spend some time on this functionality.

A) Establishment, modification, and close of an Association.

B) What parameters can be exchanged and when

C) What mechanisms are used for authentication

D) Other options

Directory Security Services

Directory SIG has reached agreements on access control and authentication as it applies to this standard. A summary of what has been agreed to is as follows:

A) Peer Entity Authentication via passwords or digital signatures

B) Integrity

C) Access control via lists

D) Relationship between authentication and access control

Security SIG

The Security SIG is a forum for security architecture, modeling, profiling, and algorithm registration issues. One area that this group will tackle is that in a full OSI compliant system, containing a multitude of OSI applications over the seven layer model, security could be almost everywhere. The SECSIG will need to profile different scenarios so that there is not a lot of duplication of security functionality. This SIG will get extremely busy as more base standards are agree to internationally. The other regional Security SIGs will also be part of this security harmonization process so that the user does not pay a steeper efficiency penalty then necessary for their secure communication needs.

Emerging OSI Security Protocols & Techniques

Ted Humphreys
XISEC Consultants Ltd., England
12th July 1992

This short note reflects on the position of current international standards being developed by ISO/IEC JTC1, in the area of OSI security protocols and techniques.

SECURITY PROTOCOLS

The two sub-committees responsible for the standardisation of OSI protocols and their security extensions are ISO/IEC/JTC1/SC21 (OSI Architecture, Upper Layers and Management) and SC6 (OSI Lower Layers). In addition, ISO/IEC/JTC1/SC18 is dealing with X.400 MHS security in collaboration with CCITT.

The current scope and status of the work in these areas includes:

Physical layer security enhancements have reached international standards status as ISO 9160. This defines the interoperability requirements for line encipherment devices, called Data Encipherment Equipment. It specifies, for each of the V.24, X.20bis, X.21bis, X.20 and X.21 physical interfaces:

- the means of exchanging session keys & initialisation variables;
- the point at which encipherment commences;
- the actions taken when error occurs.

Network layer security enhancements are defined in the ISO 11577 Network Layer Security Protocol (NLSP) standard. The NLSP specifies optional additional protocol permitting the use of cryptographic techniques to provide data protection for network layer connections and for connectionless network later transmissions. The NLSP provides security features in the Network Service as described in ISO 8348 and ISO 8348/ADI and augmented by ISO 7498/2.

The NLSP is applicable in a wide range of threat environments and may be implemented to a range of assurance levels. It can be used as a self-contained protocol within the network layer, it can be closely coupled with a SNAcP (Subnetwork Access Protocol as defined in ISO 8208) and it can be provided in such a way that operation of the protocol incurs the minimum of possible overhead. In addition the NLSP can be implemented in either an end system or an intermediate system.

The NLSP protocol has two basic modes of operation (i) NLSP-CL which provides at its upper boundary a secure connectionless network service and (ii) NLSP-CO which provides at its upper boundary a secure connection oriented network service.

Both modes can be implemented in end systems and in intermediate systems. These modes allow for the source and destination NLSP address to be optionally protected from disclosure. Both modes can be operated anywhere within the network layer such that the upper and lower layer services use the defined primitives.

The NLSP provides the same mode of service (CL or CO) at its upper and lower boundaries.

The NLSP security features are based on the use of cryptographic mechanisms and provide the following facilities:

- connection (NLSP-CO) & connectionless (NLSP-CL) confidentiality
- connection mode integrity without recovery (NLSP-CO)
- connectionless integrity (NLSP-CL)
- access control (NLSP-CO & NLSP-CL)
- data origin (NLSP-CL) & peer entity authentication (NLSP-CO)
- traffic flow confidentiality (NLSP-CO & NLSP-CL)

The NLSP makes use of the concept of a Security Association (SA) which may exist outside of a specific connectionless UNITDATA or connection. SAs are established between communicating parties and are used to protect an instance of communication (connectionless SDU or a connection). The information forming an SA are those keys and other security attributes needed to control the operation of security.

The NLSP is currently at the Committee Draft (CD) stage of development.

Transport layer security enhancements are defined in ISO 10736 the Transport Layer Security Protocol (TLSP) standard. The TLSP specifies procedures that operate as optional extensions to those defined in ISO/IEC 8073 (Connection Oriented Transport Layer Protocol specification) and ISO 8602 (Protocol Specification for providing connectionless mode transport service). This does not preclude unprotected communications between transport entities implementing 8073 and 8602.

TLSP is applicable in a wide range of threat environments and may be implemented to a range of assurance levels. It can support all the integrity, confidentiality, authentication and access control services identified in ISO 7498-2 as relevant to the transport layer.

The TLSP supports these services through the use of cryptographic mechanisms, security labeling and attributes, such as keys and authenticated identities, pre-established by security management.

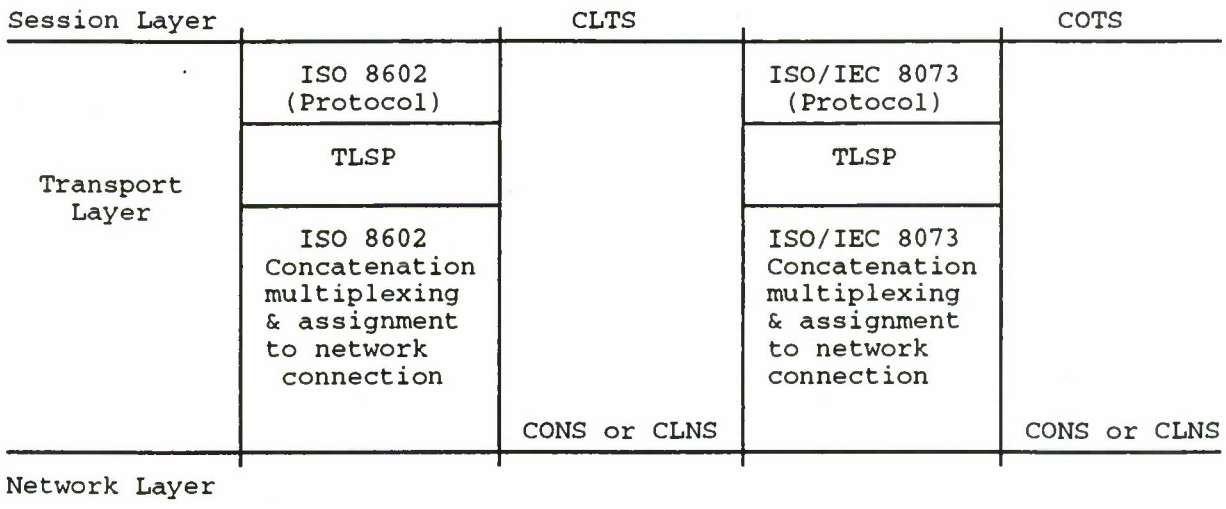
TLSP used with ISO/IEC 8073 can support connection integrity with and without recovery, connection confidentiality, access control service and peer authentication with each connection individually protected.

TLSP used with ISO 8602 can support connectionless integrity, connectionless confidentiality, access control service and data origin authentication.

The TLSP specification describes the protocol extensions for providing confidentiality and integrity data protection, in particular it defines:

- the procedures for incorporating cryptographic techniques in protocol processing,
- the minimum characteristics of cryptographic algorithms with which these procedures can be used,
- the structure and encoding of data units necessary to achieve interoperability.

The two modes of operation specified in the TLSP standard are shown in the following figure:



The TLSP is currently at the Draft International Standard (DIS) stage of development.

Message Handling Systems security features have been incorporated in the CCITT X.400 ISO 10021 standard. These features provide a secure message transfer capability within a message handling environment. The following are some of the facilities defined:

- Authentication (message origin, peer entity, etc.)
- Integrity (message content, message sequence, connection)
- Non-Repudiation (of message delivery, origin, submission)
- Confidentiality (message content, message flow, connection)

The X.400 MHS services and protocols is designed to support a number distributed messaging applications e.g. E-Mail and EDI.

This work has now been extended to cover EDI messaging security as specified in the CCITT X.435 standard. An ISO equivalent of X.435 is currently under development.

Further security enhancements to X.400, e.g. in the area of Interpersonal Messaging, are expected to be developed over the next few years.

Directory System security has been incorporated in the CCITT X.500 ISO 9594 standard. Security is provided in the form of Authentication and Access Control capability to support Director Services. The Access Control capability presently specified is limited in features and in scope, but work is currently under way to extend and strengthen this capability.

Other OSI Upper Layer security work includes (i) FTAM security which is slowly starting to be developed, (ii) TP security is planned for development, (iii) ODA security is under development and will be available shortly, and (iv) Remote Database Access (RDA) as defined in ISO 9579 is now available.

SECURITY TECHNIQUES AND MECHANISMS

The OSI Security Architecture, ISO 7498-2, identifies a number of security mechanisms and techniques that could be used to implement the security enhancements of OSI services and protocols. In particular those lower layer and upper layer security protocols described above.

Specific mechanisms include:

- Encipherment Techniques - these can be used on their own to provide data confidentiality services or in combination with other techniques to provide services such as peer entity and data origin authentication, and various integrity services.

- Digital Signature Mechanisms - these mechanisms are concerned with data appended to, or a transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and to ensure against forgery. These mechanisms can be used as part of a non-repudiation service with proof of origin or proof of delivery, and can complement a number of other security mechanisms.

- Access Control Mechanisms - these are used to determine and enforce the access rights of an entity attempting access to an OSI resource. These may be based on the use of passwords, access control lists or some non-forgeable capability.

- Data Integrity Mechanisms - are normally based on the use of cryptographic and

non-cryptographic check functions and values. These techniques can be used to provide services such as connection integrity, selective field integrity and connectionless integrity.

- Authentication Exchange Mechanisms - these may employ the use of cryptographic and non-cryptographic techniques, characteristics and/or possessions of entities and resources and hand-shaking protocols. These techniques can be used to provide services such as peer entity authentication.

- Notarisation Mechanisms - these provide assurance of certain properties (e.g. integrity, origin, time and destination) associated with the data being communicated between entities. These techniques can be used to provide services such as non-repudiation of origin or delivery.

The sub-committee generally responsible for the standardisation of Security Techniques is ISO/IEC/JTC1/SC27. The scope of SC27 covers the development of standards for security techniques such as digital signatures, authentication exchange mechanisms, data integrity and non-repudiation techniques.

In addition, SC27 is also responsible for dealing with key management techniques, non-cryptographic techniques, management guidelines, security services in general (e.g. trusted third party services for EDI) and security evaluation criteria.

Although there are a number of examples of standards for security techniques e.g. ISO/IEC 9796 "Digital Signature Mechanism with Recovery", ISO/IEC 9798 "Entity Authentication Mechanisms" and ISO 9797 "Data Integrity Mechanism", there are still a number of mechanism standards yet to be produced. SC27 will, as part of their current scope, be instrumental in such developments.

On the specific issue of algorithms used in the implementation of such mechanisms, there is available an "International Register of Cryptographic Algorithms". The standard ISO/IEC 9979 "Procedures for the Registration of Cryptographic Algorithms" defines the content and format for register entries along with the rules of submission and other administrative details for algorithms to be registered. The International Registration Authority responsible for this register is NCC (National Computer Centre) in the U.K..

USER & IMPLEMENTOR ISSUES

The above descriptions refer in the main to the development of base standards. One of the major concerns of users and implementors is the number of protocol and service options that are defined in these standards. Thus the next important step in the development process is the definition of security (sub)-profiles and implementors agreements based on these base standards. Such profiles and agreements define subsets or combinations of base standards to provide specific functions. They identify the use of specific options available in base standards. They generally provide a reduced set of options for users, purchasers and developers of IT products and systems.

Work is being carried out with the various Regional Workshops (e.g. OIW in North America and EWOS in Europe) in the development of profiles and implementor agreements to facilitate the effective use of base standards in a business environment. Work in this area has resulted in the development of an agreed OIW/EWOS X.400 MHS Security Profile. Work has also started on the profiles for lower layer security and EDI messaging.

The needs of IT users, and purchasers for security are very diverse, as are the range of threats and vulnerabilities of real operational systems. Hence other user/implementor issues include: (i) the mapping of security functionality specified in these profiles onto the right levels of protection and assurance to meet operational requirements (e.g. in the case of OSI lower layer security), (ii) the appropriate selection of mechanisms/algorithms to meet these levels of security to be implemented and (iii) the combination of security standards (protocols, services and techniques, etc.) for the development of security systems (e.g. security could be provided by more than one OSI layer via a combination of security mechanism - say TLSP options combined with
options from X.400 to provide a basis for security messaging services).

In summary, there are still a number of standardisation developments to be progressed, in particular in the area of techniques and mechanisms, before we have a complete set of building blocks for secure open systems. In addition, there are several user, purchaser and implementor issues, including those above, which focus on the implementation of security using these standardised building blocks. Hence the international community is still some years away from having fully standardised secure open systems.

Executive Summary

SECURITY LABELS IN OSI

T.C. Bartee
Institute for Defense Analyses

The security labeling of data transmitted through communications networks is complicated by several factors, among these are: 1) the labels must ride inside formats dictated by protocols, 2) at each step the labels must be capable of being moved up or down in a protocol stack.

For network usage, labels must move between user hosts, between subnets, and through intermediary devices, each of which impose differing demands. The problem is sufficiently sending labels out-of-band and then "tokens" in-band and partitioning security labels and sending them through different layers.

Fortunately, both standards organizations and users have been aware of the potential problems which can arise if protocols and labeling mechanisms are not adequate. Standards organizations such as NIST have been generous with their accommodation of user viewpoints through workshops and committee user representations. Still, user communities must realize the extent and complexity of the problem and continue to work for protocols, standard labeling formats, and label registration which will permit adequate labeling.

This presentation will take the following positions:

1) The security labeling mechanisms provided in standards for protocols must be adequate for both present and future usage. These mechanisms must allow for the many strategies for passing security information which may arise.

2) Security labels must be carefully worked out in user communities. The labeling design must be adequate and allow for growth. Those familiar with the data element naming problem will recognize the problems inherent in communicating using labels which have not been coordinated between user communities. Once computers and systems are full of labels, it would be very difficult and expensive to make changes. Translation is sometimes difficult and sometimes nearly impossible.

3) The DoD has emphasized classification levels as a guide to the "level" of information and to the degree of protection to be accorded data. While this categorization is an important one (it is certainly historically important), much data at the "lower" level of (Government) sensitive must be protected to the same extent as upper level data. A compromise of Law Enforcement Sensitive information, for example, can endanger the lives of officers and informants.

4) Data throughout the Government is more and more being divided into categories and the data must be controlled so that only those entitled to data in a particular category receive it. The compartments in DoD systems offer examples of such categories and so do such Counter Narcotics categories as FBI - Protected Source, Coast Guard - Suspect Vessel, etc. Undesired disclosure of such information can endanger operations and personnel (disclosure of radar track information can, for example, reveal radar locations.) For commercial concerns, results of studies, customer and personnel files, product development data, etc. must all be protected.

5) For category information, there are several other important considerations. Sometimes data is originally distributed to a limited group and further dissemination is only permitted with permission from the originator. In order to observe this rule (called ORCON, third party, etc.) it is necessary to identify the originator or, if data from several sources is merged, the originators. This, plus other factors, can cause markings to be context dependent in ways that will be shown and further complicates marking.

Status information on current labeling standards will be presented as will some examples of how these standards are being used and insights into what are the current problems.

Panel:
Challenges Facing Certification and Accreditation Efforts of the Military Services

Chair: LTC Ron Ross, USA

Panelists: Larry Merritt, AFCSC, San Antonio, TX
Robert Zomback, CECOM, Fort Monmouth, NJ
John Mildner, NESSEC, Washington, D.C.

Abstract. The purpose of this panel is to inform the Department of Defense (DoD), Intelligence community, and civilian agencies and their contractors of initiatives to implement the policies of DoD Directive 5200.28 in the United States armed services. This panel will consist of a representative from each of the services. Discussion will focus on initiatives to address some of the challenges and issues in system certification and accreditation (C&A).

BACKGROUND

In recent years, there has been a shift in perspective of automated information system (AIS) security from viewing it as a number of independent, loosely coupled disciplines to a more cohesive, interdependent collection of security concerns. The current environment of declining resources and rapid advances in technology have demanded changes in evaluating the security posture of systems. These changes are necessary to reduce fragmentation and to ensure consistency and compatibility. In addition, an ever-changing threat environment requires a more efficient, integrated view of INFOSEC, which includes emissions, communications, computer (hardware/software/firmware), administrative, personnel, and physical security.

DoD policy states that any AIS that processes classified, sensitive unclassified, or unclassified information must undergo a technical assessment and management approval before it is allowed to operate. The technical assessment establishes the extent to which the system meets a set of specified security requirements for its mission and operational environment. The technical assessment and management approval processes are called certification and accreditation, respectively.

The Defense Information Systems Security Program (DISSP) has identified the insufficiency of certification and accreditation guidance for many environments, and its inconsistency across DoD components, as major problems with current DoD policy. The use of AISs within all aspects of DoD operations, the dynamic organization of systems, and the exchange of information among systems point to the need for uniform guidance when certifying and accrediting AISs. DISSP has identified the development of uniform accreditation criteria as one of the most important near-term tasks required to provide an integrated, cost-effective DoD information systems security program.

Described below are organizations in the services which help to conduct technical assessments supporting the C&A process. Also described are the challenges and issues they face.

ORGANIZATIONAL STRUCTURE AND RESPONSIBILITIES

U.S. Air Force

The Air Force consolidated its INFOSEC resources in 1988 at the Air Force Cryptologic Support Center (AFCSC), Kelly Air Force Base, San Antonio, Texas. The organization responsible for managing the INFOSEC Program at AFCSC is the Communication-Computer Systems Security Directorate. The mission of this organization is to manage the Air Force Communication-Computer Systems (C-CS) Security Program. The charter of this organization includes:

- Providing program management and implementation planning to establish and provide an effective Air Force C-CS security program
- Managing the security activities of Air Force, MAJCOM, and direct reporting unit (DRU) C-CS, to ensure uniform and effective program implementation
- Providing security education, training, and awareness to all functional levels and operational activities within the Air Force and its contractors
- Developing policies, procedures, and criterion, as required, to ensure compliance with established Air Force and national level objectives
- Developing security architectures in support of the overall Air Force C-CS systems architecture
- Providing C-CS security research and development (R&D) in conjunction with Air Force Systems Command (AFSC) and other DoD agencies
- Providing on-site technical system evaluation during system design reviews, systems integration, and test and evaluation
- Providing for COMPUSEC product evaluation and assessment for security products sponsored by Air Force users with Air Force-wide applications in conjunction with the National Security Agency
- Acting as Air Force executive agent for managing C-CS security support to communications and weapons system acquisition/implementation programs

U.S. Army

The Army consolidated its INFOSEC resources in support of tactical systems in the mid-1980s at the Communications and Electronics Command (CECOM), Fort Monmouth, New Jersey. The organization responsible for managing the INFOSEC Program at CECOM is the Information Security Division (ISD). The mission of this organization is to conduct research through development prove-out and production directed at the application of security technology, equipment, and concepts for tactical and strategic/sustaining base Army communications and automation systems. The charter of this organization includes:

- Conducting research through development prove-out and production directed at the application of security technology, equipment, and concepts for tactical and strategic/sustaining base Army communications and automation systems

- Acting as level II project manager and providing post deployment field support, when directed
- Providing systems security expertise to Army Program Executive Officers (PEOs), Program Managers (PMs), and other R&D centers
- Representing the Army in joint service security working groups
- Providing the technical support for Special Operations Forces (SOF) and drug law enforcement agencies.

U.S. Navy

The Navy is proposing to establish an organization similar to those of the Air Force and Army. The Space and Naval Warfare Systems Command (SPAWARSYSCOM), located in Washington, D.C., is proposing to set up an INFOSEC Organization. The INFOSEC Organization will provide centralized management of SPAWAR INFOSEC resources for Department of the Navy (DON) customers, and will be the primary customer point of contact. Additionally, this proposal includes the formation of an INFOSEC Engineering Center (IEC) at the Naval Electronic Systems Security Engineering Center (NESSEC), located in Washington, D.C., to perform systems security engineering functions for DON systems. The charter for the Navy's IEC has not been approved yet, but it will have similar responsibilities to the INFOSEC centers located at AFCSC and CECOM.

SECURITY CHALLENGES AND ISSUES

This panel will not focus its discussion on the infrastructure of the INFOSEC organizations of the services. Rather, it will address a variety of challenges and issues that impact the implementation of C&A. Based on input from representatives throughout DoD, two areas of concern that were identified by the DoD Certification and Accreditation Working Group, which is sponsored by NSA, are policy and process.

Policy

A proliferation of policy at the national, DoD, service/agency, and lower levels makes it difficult for security personnel to keep up with changes in policy, and to be aware of all the applicable policies for a given system. The problem increases when service/agency systems are interconnected. In those cases, the policies relevant to all involved components may then be applicable. Due to the rapid advancement of technology and the streamlining of procurement efforts due to resource limitations, a reexamination of current policy, to include C&A policies, is necessary. The following are some of the most important issues regarding current policy:

- Policy not applicable to range of systems
- Insufficient or nonexistent certifications and accreditations
- Delegation of accreditation authority
- Acceptable level of risk
- Certification boundaries

Process

Certification is conducted in support of the accreditation process as part of the risk management process. Risk management is the total process of identifying, measuring, and minimizing uncertain events affecting resources. It includes risk analysis, cost-benefit analysis, countermeasure selection, security test and evaluation, countermeasure implementation, and systems review. Enclosure 3 to DoD Directive 5200.28 mandates a risk management program for each AIS to determine how much protection is required, how much exists, and the most economical way of providing the needed protection. The management commitment to a comprehensive risk management program must be defined as early as possible in the system life cycle. In scheduling risk management activities and designating resources, careful consideration should be given to C&A goals. Some topics of concern in refining the C&A process include:

- Streamlining the process to save time and money
- Implementing consistent methodologies
- Ensuring C&A is adaptable to existing systems
- Recertifying systems
- Ensuring product evaluations feed C&A

SUMMARY

This panel does not attempt to elaborate on all of the issues that may exist, but identifies the ones that INFOSEC organizations have to deal with on a daily basis. The panel will address these issues and current initiatives that are underway to resolve the inconsistencies that may exist with policy and the implementation of the C&A process.

Acknowledgements

The moderator would like to thank ENS Susan L. Mitchell, USN, and Arthur R. Friedman, The MITRE Corporation, for organizing this panel and writing this position paper; and the members of the DOD Certification and Accreditation Working Group for their ideas and contributions in identifying the security issues and challenges discussed during the panel.

DOMESTIC PRIVACY: ROLL OF HONOR AND HALL OF SHAME

Wayne Madsen[1]

P.O. Box 302
Moorestown, New Jersey 08057

Within the past year, several organizations and people have stood out with regard to the issue of data privacy in the United States. Fortunately, some have distinguished themselves in promoting the concept and cause of privacy of personal information. Others, however, have done their best to impede data privacy. The Roll of Honor and Hall of Shame distinguishes the clear lines that are being drawn in the battle for privacy in the United States.

ROLL OF HONOR

The Public Utilities/Services Commissions/Boards of Massachusetts, Kentucky, Connecticut, Nevada, and five other states. These commissions have been instrumental in holding the telecommunications carriers responsible for offering free per-line blocking for caller ID services. Per-line blocking is of special importance for organizations such as battered spouse and child abuse hotlines in which anonymity is key. It also protects the privacy of individuals who call direct marketers and other vendors for information on products and services but want to maintain their anonymity until they decide to make a purchase or procure a service.

The Computer Professionals for Social Responsibility (CPSR). The CPSR has led the way among public advocacy groups in the privacy battle. Undeterred by well-funded industry lobby groups and powerful congressional opponents of privacy legislation, CPSR has fought to elevate the awareness of members of Congress, industry, the Federal Government and individual citizens of the dangers posed by unrestricted use of information technology in government and industry.

Senator Ernest Hollings, Chairman of the Senate Commerce Committee, who had the foresight to recognize that the FBI was trying to pull a fast one by attempting to attach a telecommunications eavesdropping rider to the 1992 FCC Authorization Bill for which his Senate Committee had oversight responsibility.

Representative Pete Stark of California for sponsoring the Prescription Drug Records Privacy Protection Act of 1992. This bill protects consumers against pharmacies and doctors accessing their pharmacy records contained in databases without the consent of the patient concerned. Several pharmaceutical firms were interested in gaining access to patient data bases to determine, among other things, the income levels of

---

[1] The views expressed herein are the author's own and do not necessarily represent the opinions of his employer or his publishers.

patients based on their types of prescriptions.

Representative John Conyers for sponsoring the Human Genome Privacy Act (HR 2045). This act would prohibit disclosure of personal genetic data without the data subject's consent and also allow for corrections and deletion of erroneous genetic data by the data subject.

Representative Bob Wise of West Virginia for once again sponsoring the Data Protection Act. This act would create a Federal Data Protection Board, which although non-regulatory in nature, would be a first step toward aligning the United States with international norms and standards on the protection of personal data.

HALL OF SHAME

The Federal Bureau of Investigation for once again showing an amazing lack of concern for the right of individual privacy. Its repeated attempts to have Congress bless its plans to engage in "turn key" electronic eavesdropping of the national digital telephonic network stand in the forefront of the assault on privacy. This assault took two forms: 1) a desire to have the technical cooperation of the telecommunications carriers to permit the FBI remote eavesdropping capabilities and 2) an attempt to regulate the use of encryption systems in the United States for which the government does not possess the decrypt capabilities. Also the FBI is singled out for its determination not to entertain any outside scrutiny over privacy controls for NCIC 2000 and the DNA Fingerprint Laboratory.

The Los Angeles Police Department and its former Chief, Daryl Gates for permitting a secret unit, the Organized Crime Investigation Division (OCID), to engage in eavesdropping activities of political, sports, journalistic and entertainment figures. OCID scanned a multitude of confidential personal data files to include the data in its own files. Also added to this list is current LAPD chief Willie Williams for not permitting disclosure of personal intelligence files to the data subjects concerned.

The Direct Marketing Association (DMA) for its myriad activities all designed to gut data protection efforts in the US and abroad. These activities are all based on the DMA's contention that Americans are afforded enough in the way of personal data protection by the DMA's much touted "opt out" clause, a proviso that individuals may elect to have their names removed from direct mailing lists. The DMA has sought to sell this approach to other countries in order to either influence their proposed data protection legislation or amend current legislation in order to water down its more stringent regulations for controlling the handling of personal data by the private sector.

The National Institutes of Health for its lack of concern for privacy protection for data accumulated as part of the Human Genome Project. Genetic data collected by NIH is of prime interest to the insurance companies and not enough is being done by NIH

to ensure that this data does not fall into their hands.

The Department of Justice for failing to adequately investigate the death of journalist Joseph Daniel ("Danny") Casolaro in 1991. Mr. Casolaro had been investigating the alleged theft of PROMIS, a powerful court case tracking relational data base program, by Justice Department agents from Inslaw, Inc., its subsequent illegal duplication and transmittal to the Central Intelligence Agency and its illegal export to the security services of other countries like Syria, Iraq and Libya.

The US Department of State for its lack of appreciation for data protection initiatives on the international stage and its blatant attempts to scuttle certain initiatives (this especially included allowing the Direct Marketing Association and the US Chamber of Commerce to dictate US data privacy policy in international fora). International initiatives to enhance personal data protection include the UN General Assembly Resolution, the EC Directive, the OECD Security Guidelines (watered down by the State Department), GATT negotiations (Subsection G of Article 14 of GATT's Uruguay Round Accord), and proceedings of the International Data Protection Commissioners Conference. Also for attempting to dissuade private US citizens from participating and voicing their opinions on the lack of a US data privacy commitment in international settings.

Ross Perot for his alleged practice of maintaining personal dossiers on his political and business rivals including the President of the United States. Under standard European norms of data protection legislation such file keeping practices would be illegal. These practices also throw into question the data privacy ethics maintained by Perot's former and current data processing enterprises.

Executive Summary

Panel: Health Issues Program

Gerald S. Long, Chair

Harrison Avenue Corporation
812 Downs Drive
Silver Spring, MD 20904

(301)622-3177

The health care industry covers a broad spectrum of information security needs and challenges. It is an industry which historically has used the paper medical record as the communal document of the health care team; physicians, nurses, health care specialists, technicians, educators, researchers, and administrators. A "white coat" generally being the admissions pass to the medical record.

With computerized records, established ways are being challenged and "need-to-know" and special regulations limit access by members of the health care team to all or parts of the medical record and to other health care documents and information. Other regulations control the release of sensitive information, unauthorized modification of medical record information, as well as damage to, or interruption of, medical computer services.

This program consists of 3 sections: a paper titled: *Analysis of the Applications Layer Security Requirements of a Medical Information System*; 4 points of view concerning sensitive health care data, portable data carriers, control of "need-to-know" access to sensitive data, computer-based communications and automation of patient data, and the implementation of technology to implement patient care utility services; and a panel discussion prompted by questions from the chairperson and the audience. The panel members have agreed to stay beyond he 5:30 p.m. scheduled completion time to discuss the implications of a national patient database.

Point of View

The Benefits of Smart Card Technology in the Health Industry

Peter M. Fallon
National Sales Manager
Toshiba American Information Systems

Today hospitals and health care providers are coming under increasing pressure to provide more cost efficient and better health care to their patients. They are struggling to operate with a paper based system that must be modernized with current technology. More and more rules are being put in place that demand the latest computer technology so that hospitals and other providers will be able to operate more efficiently for their patients' benefit as well as for their own bottom line.

These new methods include the latest communication as well as up to date computer automation. Smart cards play a small, but important, part of that necessary, but very important, system updating process. All of these applications are based on the secure software and hardware features that are built into today's smart cards. As with most industries today, new secure measures must be inherent in any new system that will defeat any measures that may be put into use to mount a hardware or software attack on the system.

Some of the applications we will discuss are as follows:

- Patient cards that would hold the following information

  Name, address, etc.
  Chronic medical problems.
  List of current medications.
  Last hospital visit.
  List of most important past operations.
  Allergies.
  Where to go to access more data about this patient.

- Use the card to order tests as well as patient billing and tracking within the hospital.

  Lab tests.
  X-rays.
  Drugs.

- Financial information.

  Current insurance plan.
  Deductible.
  Medicaid information.
  Medicare information.

- Tickler for patient health maintenance.

- Special confidential note section for doctors.

  Lab or x-ray findings.
  Prescription information.
  Other providers involved with the patient.

These are a few of the many applicable smart card applications.

Point of View

National Health Card

B. Bahramian,
President
Beta Management Systems, Inc.

The ever increasing costs of health care have become a major concern of every American. Lowering the cost of health related services is imperative if the economy and reasonable standards of health care are to survive. Today's advancements in technology can help in generating tremendous savings in the health care administration. However, in virtually all cases, these benefits are realized as increased profits to the carriers and providers. The successful plan should be structured in a manner to assure that these savings are realized and applied in the form of reduced premiums to the individuals, companies, groups, or governmental entity that is paying for the health care services.

Over utilization and redundant tests or procedures are the major elements in driving the health care costs to an astronomical level. Falsifying the identity of recipient is another factor which is costing the nation billions of dollars every year. All this can be reduced by introduction of a "National Health Card," with a built-in micro-chip which would carry the history, physical and clinical details as well as insurance coverage of individuals and the concealed photo ID of the beholder. The Cards may be used at hospitals, pharmacies or even in ambulances. This vehicle can help to integrate the society to receive the same treatment and services from providers, no matter who has paid for the premium.

The solution in providing a satisfactory health care service is not in socializing medicine, or dragging the government into providing the desired services. The answer is people's participation in a national plan, organized at local levels, under a Health Commission Board, similar to the management of utility services.

Point of View

The Optical Card as a Portable Medical Record

Stephen D. Price-Francis, Manager
Market Development
Canon Canada, Inc.

The Portable Medical Record, in the form of the "personal health card" or "patient card," is seen by many observers as a much-needed catalyst for change in the delivery of health care services.

This vision foresees each individual carrying a complete medical history in a convenient and portable form - typically the size and shape of a credit card. The information stored on this record would duplicate and consolidate (but not replace) records currently stored in a variety of locations, such as physicians' offices, hospitals, clinics, pharmacies, and so on.

This portable medical record would be always available, accurate and up-to-date and thus an invaluable tool for diagnosis and treatment, in both routine and emergency situations, and for referral. The record would be automatically updated during each health care encounter and carried away again by the patient.

A topic of fundamental importance is that of privacy, both of the patient and the provider, and confidentiality of medical records. The patient-borne record represents, in effect, the ultimate distributed database, but with the advantage that real-time access to the information is available only with the informed consent of the patient (except, probably, in the case of emergency information). This is in contrast to the acknowledged risk of computer network penetration by the determined "hacker" who, if successful, could have access to thousands, or even millions, of clinical records.

The data on the patient care will be protected from unauthorized access by encryption, with varying levels of access provided to different provider disciplines. Fraudulent copy or creation of a card can be prevented by a special technique of card "fingerprinting."

The principal benefits of the system are expected to be better quality care and the containment of escalating health care costs. The application of this advanced technology will provide real advantages to the users and providers of the service, and deliver important relief to those paying for the system.

This paper will consider why and how these benefits will be derived and look at an ideal technical solution to the problems posed.

Point of View

Patient Data Confidentiality in the Health Care Environment

Marc Schwartz
Director of the Medical Services
Summit Medical Systems, Inc.
(612) 473-3250

For the majority of us, the most private part of our lives is our health. For those who are of ill health and come in contact with our health care delivery system, we enter into a world that is far from ideal when it comes to protecting that privacy. The recent forced disclosure by tennis star Arthur Ashe of his HIV status, demonstrates the anxiety we all have about the discussion of any details of our state of health. The potential embarrassment we must deal with and the impact on friends, family, lifestyle and even employment makes such a disclosure a difficult task indeed.

Upon entering the health care environment, we find a world that is filled with such confidential information, a necessity to provide appropriate diagnostic and treatment options to the patient, yet with a system for protecting that data that is less secure than that which banks and airlines use for their least sensitive information. The typical point of storage of this information is the patient record. All of his or her past medical history, information about family, personal contacts, diagnoses, treatments, medications and personal patient-physician communications are contained in this important set of documents.

For medical and legal reasons, each personal physician, specialist, hospital department, clinic, urgent care center, same day surgery center or other health care professional or facility must keep records of all contact with the patient. Thus a patient who has a chest x-ray after surgery, will find copies of the dictated interpretation in the radiologists office, the hospital x-ray department, the surgeon's office and likely a copy forwarded to his or her personal physician. The same situation occurs for many diagnostic or therapeutic treatments. This does not take into account "routine" conversations about patients that can take place anywhere from the patient care areas to the doctor's locker room and cocktail parties at which physicians and other providers can be overheard by people totally unrelated to that patient's care.

Another point of storage for this data is the clinical database. As opposed to the patient record, this system stores a sub-set of related information on many patients, usually including patient identifiers, thus facilitating statistical analysis of a specific group of patients, such as all patients undergoing open-heart surgery. Most academic institutions and many medical societies are involved in some form of research study, as are most medical device manufacturers in the process of bringing products to market or for post-market surveillance now required by the Safe Medical Devices Act of 1990. With increasing emphasis on cost-containment, many "*outside*" peer-review organizations, government agencies such as HCFA and insurers are utilizing this data to compile patient outcome based policies for anything from reimbursement criteria to physician and hospital standards of practice for quality assessment and professional credentialing.

Each of these locations provides a possible "*covert channel*" for data leakage to unauthorized parties. The lack of standards for data storage, security and integrity provides a ripe opportunity for illicit entry into this mass of private data. Who is ultimately responsible for the protection of this data? What methods are in place to maintain its integrity? Can the patient review this data to make sure that it is accurate, just as one can review their credit history and request correction of inaccurate data. Would a patient even be in a position to know if such information is inaccurate without appropriate medical knowledge? Who is authorized to review this information and what access controls are in place to insure the procedure? What, if any, "*need-to-know*" policies are in place at each point in the chain of data repositories? Under what circumstances is the patient's authorization required for release of data to outside parties? What professional guidelines are in place to provide for ethically, if not legally, binding awareness on the part of the health care community? What type of protection is provided to the providers themselves, to protect these people from the invasion of their privacy? In this age of increasing competition in the health care environment, where the financial success of a provider is based on getting as many patients into their institution or group as possible; where mergers and acquisitions of hospitals and clinics are taking place almost daily; where medical advertising is becoming the rule; will providers engage in "*industrial espionage*" to gain covertly, information about competitors or their clientele (the patient) that can be of use for competitive advantage? How will the computerization of these records impact upon the security afforded this information?

Executive Summary

Panel: Information Technology (IT)Security Requirements

Mr. Dennis Gilbert, Chair,
National Institute of Standards and Technology

N. Lynch, *National Institute of Standards and Technology*
Ms. Sadie Pitcher, *Department of Commerce,*
Needs Study Working Group
Dr. W. Maconochy, *National Security Agency,*
FISSEA Chair
Ms. Marianne Swanson, *National Institute of Standards and Technology,*
CSBBS SYSOP

Federal agencies and other organizations are faced with a variety of requirements concerning the protection of sensitive information and the resources used to store and process that sensitive information. These information technology (IT) security requirements derive from a number of sources. Among these sources are legal and regulatory responsibility; Privacy Act and other privacy concerns regarding employees, clients, and customers; fiduciary and custodial responsibility; national security; desire for public, client, and customer confidence; good management and business practice; fear of fraud and embezzlement; fear of litigation and increased regulation; and ethical concerns. The rapidly changing information technology environment and a number of short term and long term trends raise new IT protection concerns and challenges.

What are these needs and requirements? How do we determine what help agencies and other organizations need to address these requirements? What resources are available to help and what are the means to focus resources appropriately?

A number of recent efforts attempt to address these questions. In one of these, the National Institute of Standards and Technology (NIST) conducted a study to help understand and document what federal

agencies need to satisfy their information technology (IT) security requirements. The study involved interviews with identified agency staff and the use of an IT security needs assessment survey. Respondents were asked to identify their IT security needs from a list of candidates and indicate their importance and immediacy. Other federal, private sector, and professional organizations completed the needs assessment survey form. A working group representing the federal and private sectors participated in the design of the study and the validation of study results. These results will be used by NIST to better gear its programs to help agencies satisfy those needs and to plan for the effective use of NIST resources.

Other recent activities give us additional insight. These include:

> Computer security curriculum development efforts by the Federal Information Systems Security Educators Association (FISSEA)
>
> OMB, NIST, and NSA agency assistance visits made in accordance with OMB Bulletin 90-08 (and the results of the NIST and NSA Computer Security and Privacy Plans review effort done in accordance with OMB Bulletin 88-16)
>
> A NIST study of technical information protection methods used in computers or application systems in government and industry. (The study involved in-person discussions with key persons in 17 federal agencies, 10 commercial organizations, and one state government, between March and June 1991. Approximately 120 people were interviewed. The study provided input to NIST's Minimum Security Requirements/Federal Criteria efforts.)

This panel session will examine these activities and related activities. Also discussed will be: methods of IT security needs assessment that can be used by the audience; and resources available to address identified IT security needs.

INTERNATIONAL DATA PRIVACY: ROLL OF HONOR AND HALL OF SHAME

Wayne Madsen[2]

P. O. Box 302
Moorestown, New Jersey 08057

Within the past year, several organizations and people have stood out with regard to the issue of data privacy throughout the world. Fortunately, some have distinguished themselves in promoting the concept and cause of privacy of personal information. Others, however, have done their best to impede data privacy. The Roll of Honor and Hall of Shame distinguishes the clear lines that are being drawn in the battle for privacy on the international stage.

ROLL OF HONOR

Louis Joinet, of the UN Commission on Human Rights, for his tireless efforts to push data protection on a pan-European and international scale through the United Nations, Council of Europe and the European Community.

Joachim Gauck, the Commissioner for Stasi records of Germany, for streamlining the process of disclosure of personal files to victims of Stasi surveillance in 1992.

Askar Akayev, the President of the Republic of Kyrgyzstan for championing civil and political liberties including privacy rights in lands that were once the domain of Genghis Khan and are on the doorstep of the anti-democratic masters of Beijing.

The governments of Poland, Lithuania, Latvia, Estonia, Czech Lands, Hungary, Slovenia and Bulgaria for bringing their secret security services under strict presidential control and legislative oversight and refusing to grant exemptions from data protection/privacy legislation that are either in effect or proposed.

Privacy International and its Director-General, Simon Davies for organizing privacy advocates on a truly international scale. PI is taking on powerful privacy intruders in areas where they have for so long commanded almost unchallenged supremacy.

The Government of New Zealand for being the latest nation to adopt a Privacy Act, the Privacy Commissioner Act of 1991.

The Government of Japan and its Ministry of Posts and Telecommunications for

---

[2] The views expressed herein are the author's and do not necessarily represent the views of his employer or his publishers.

extending privacy guidelines to the telecommunications industry. These guidelines are in conformance with the OECD Guidelines Governing the Protection of Privacy and Transborder Flows of Personal Data of 1980.

Prime Minister John Major of the United Kingdom for trying to overhaul Britain's antiquated system of maintaining an inordinate amount of secrecy of government files. Major has championed a Citizens' Charter which grants British citizens freedom of information and access rights to previously classified government data.

The newly democratized nations of Africa for promoting civil and political liberties including the right of privacy in countries that have never experienced enforceable constitutional provisions for individual rights and liberties. These nations and their new democratic leaders include Zambia (Frederick Chiluba), Ethiopia (Meles Zenawi), Benin (Nicephore Soglo), Sao Tome and Principe (Miguel Trovoada), Cape Verde (Antonio Monteiro Mascarenhas), Mali (Alpha Oumar Konaré), Congo (Andre Milongo), Rwanda (Dr Dismas Nsengiyaremya), and Togo (Joseph Kokou Koffigoh).

The Data Protection and Privacy Commissions of the UK, Isle of Man, Ireland, Jersey, Guernsey, Finland, France, Sweden, Denmark, Faeroes, Germany, Iceland, Norway, Austria, Belgium, Netherlands, Portugal, Luxembourg, Japan, Australia, New Zealand, Canada, Israel, Quebec, Ontario, New South Wales, Slovenia, Hungary, Hesse, Rhineland-Palatinate, Saxony, Lower Saxony, Saxony-Anhalt, Thuringia, Bremen, Berlin, Bavaria, Hamburg, Saarland, Brandenburg, Mecklenburg-West Pomerania, Baden-Wurttemburg and Rhineland-Westphalia, the Commission of the European Communities, Council of Europe, and the Organization for Economic Cooperation and Development for their international cooperation on furthering the goals of data protection and leading the way for those nations that have yet to pass data protection legislation.

HALL OF SHAME

The European Parliament's Committee on Legal Affairs and Citizens' Rights*** for

---

***Including its members: Chairperson Franz Ludwig Graff von Stauffenberg (Germany, Christian Social Union), First Vice Chairperson Marie Claude Vayssade (France, Socialist Party), Second Vice Chairperson Willi Rothley (Germany, Social Democratic Party), Rapporteur Geoffrey Hoon (United Kingdom, Labour Party), Giorgios Anastassopoulos (Greece, New Democracy), Juan Maria Bandres Molet (Spain, People's Left), Rinaldo Bontempi (Italy, Party of the Democratic Left), Carlos Maria Bru Puron (Spain, Spanish Workers Socialist Party), Patrick M. Cooney (Ireland, Fine Gael), Michael N. Elliott (United Kingdom, Labour Party), Alexander C. Falconer (United Kingdom, Labour Party), Manuel Garcia Amigo (Spain, Popular Party), Johanna Christina Grund (Germany, Republican Party), The Lord Inglewood (United Kingdom, Conservative Party), James L. Janssen van Raay (Netherlands, Christian Democratic Appeal), Francisco Antonio Lucas Pires (Portugal, Social Democratic Center Party), Anne Carolyn B. McIntosh (United Kingdom, Conservative Party), Nora Mebrak-Zaldi (France, Socialist Party), Manuel Medina Ortega (Spain, Spanish Workers Socialist Party), Christine M. Oddy (United Kingdom, Labour Party), Giorgios Saridakis (Greece, New

watering down the proposed EC Directive concerning the protection of individuals in relation to the processing of personal data.

The Attorney General of Australia for not addressing privacy concerns in the creation of the Law Enforcement Access Network (LEAN), exempting LEAN from the provisions of the Australian Privacy Act of 1988, and providing virtual unrestricted access to some 40,000 users.

Prime Minister Brian Mulroney for proposing the combination of the Federal Privacy and Information Commissions. Such a move would weaken both offices and have a detrimental effect on data privacy of Canadians.

The Royal Canadian Mounted Police (RCMP) for placing a paid informant in the highest echelon of the former Parti Quebecois government of Quebec. The informant, a Minister for Intergovernmental Affairs, provided confidential information to the RCMP between 1975 and 1977.

The government of President Carlos Salinas de Gortari of Mexico for permitting illegal government surveillance of the Mexican National Commission of Human Rights and ignoring abuses of individual rights in the states of Sonora, Baja California, Morelos, Veracruz and Guanajuato. Added to this list is the US Administration for not pressuring Mexico to guarantee personal privacy as part of the North American Free Trade Agreement (NAFTA), especially with regard to the protection of data on US citizens processed by data centers in Mexico.

President Alberto Fujimori of Peru for imposing a dictatorship on Peru and giving new and widened powers to the National Intelligence Service (SIN) of Peru. SIN is piling up personal dossiers on opponents of Fujimori's illegal regime, including members of the dissolved Peruvian Congress, the dismissed members of the Supreme Court and human and political rights activists.

The Government of the People's Republic of China for continuing its abusive methods of maintaining personal dossiers on much of the Chinese population, for using these files for repression of Tibetan nationalists and Chinese democratic forces, and using surreptitious computer eavesdropping methods to infringe on the privacy of Hong Kong citizens' financial and other business affairs.

The government of India for continuing the dossierization of political and religious groups within the country that are opposed to Indian government policies, including

---

Democracy), Lode J. C. van Outrive (Belgium, Flemish Socialist Party) and Anthony J. Wilson (United Kingdom, Labour Party). While support to water down the EC Data Privacy Directive was not too surprising from the member representing the German Republican Party, a neo-Nazi group, the support from moderates, socialists and moderate conservative parties was probably the result of very active lobbying from the American Chamber of Commerce, US magazine publishers, US and European direct marketing groups and US and European computer vendors.

groups fighting for self-determination.

The government of Indonesia for its repressive data base program known as Penelitian Khusus which is aimed at identifying and vetting opponents of the regime and denying them their civil and political rights.

The government of Israel for not applying the provisions of the Privacy Act of 1981 to the Palestinian residents of the West Bank and Gaza. Palestinians are required to carry bar-coded identity cards and their personal data is stored on government IBM mainframes. They have no right to inspect their data nor seek correction or deletion of erroneous or irrelevant data.

Western computer firms for selling sophisticated data base management systems and smart card technology to countries wishing to use such systems for population surveillance and control.

INTERPOL for not addressing the problems associated with developing an automated international criminal information system with input and output to national criminal information systems like NCIC2000, PNC2, CPIC, LEAN, Schengen Information System, INPOL, etc. and in failing to resist US attempts to computerize millions of INTERPOL files without extending required privacy provisions to automated criminal data bases.

The German media for its irresponsible actions with regard to Stasi files. Several German newspapers and magazines have been too quick to publish the names of East German informers, and in so doing have destroyed many promising careers and have driven some people to suicide.

The government of Iraq for operating an intelligence network that spied on Kurds abroad. Extensive files were maintained on Kurds living in North Dakota, Detroit, Miami and Washington, DC. Added to this list is the US Justice Department for neither warning nor preventing such activities against Kurdish-Americans and Kurds resident in the United States because of the now well-known pro-Iraq "tilt" of the US Administration at the time Iraqi surveillance was being conducted.

The government of Vietnam for its surveillance activities of Catholics and human rights activists. The Vietnamese intelligence agency, Quan Bao, operates one of the last communist secret police apparatuses.

Executive Summary

Multilevel Security (MLS) Prototyping and Integration: Lessons Learned and DoD Directions

C. West, Chair,
Defense Information Systems

This panel will provide an overview and some focused insights into the MLS prototyping and integration efforts in DoD. A DoD MLS program was established in January 1990 with three major functions.

- to plan and coordinate DoD MLS projects and initiatives
- to develop and evaluate generic solutions at designated testbeds
- to provide engineering assistance to implement MLS capabilities

The panel will focus on the specific tasking of the DoD MLS program relative to testbeds:

- Using MLS testbeds to assess and integrate products
- Developing tools, techniques and integration methods
- Migrating capabilities from testbeds to operational systems

DoD has been involved with an MLS Testbed at USTRANSCOM since 1988. The panel will cover lessons learned from the MLS Testbed at USTRANSCOM and service and agency testbeds. Included will be discussion of a new NSA initiative to expand the evaluation process from products only to security profiles for integrated sets of products and guidelines that NSA will be developing that have been influenced by the testbed efforts. The status of current plans to establish a DoD MLS Central Testbed will be reviewed. The purpose of the DoD MLS Central Testbed is to focus on generic solutions, provide a development facility for support of the CINCs and other high priority DoD users, assess products and advance new approaches for DoD wide applicability. The current and planned initiatives of the DoD MLS Central Testbed and service and agency MLS testbeds will be covered by the testbed managers. An index of available lessons learned documentation will be provided to attendees. These include surveys and assessments of MLS workstations and MLS local area network alternatives.

Executive Summary

Workshop: New Security Paradigms

Hilary Hosmer, Chair
Data Systems, Inc.

On Tuesday, October 13 (2:00p.m. - 5:30 p.m.) and Wednesday, October 14 (9:00a.m. - 12:00p.m.), Ms. Hilary Hosmer, President, Data Security Inc, will chair a workshop focusing on new security paradigms. Shifts in paradigms, our fundamental models or views of reality, are like earthquakes. They disrupt the status quo, destroy outdated ideas, and open the way to new possibilities. The interoperability and flexibility users require in trusted systems may exceed the capability of the current TCSEC paradigm. Yet, there is no superior alternative in sight. Conference attendees are invited to share their ideas and perspectives for new security paradigms in a creative and constructive workshop which is half presentation and half discussion.

The Tuesday program is based on the best efforts of the **New Security Paradigms Workshop** (*Little Compton, Rhode Island, September 22-24, 1992*). The New Security Paradigms Workshop papers will be edited by Dr. V. Ashby, The MITRE Corporation and published by the ACM SIGSAC. The papers presented during Wednesday's program are refereed by the National Computer Security Conference.

Tuesday October 13, 1992

2:00 - 2:05	*Introduction* **H. Hosmer**, *Data Security, Inc.*
2:05 - 2:25	*A New Paradigm for Trusted Systems* **Dr. D. Denning**, *Georgetown University*
2:25 - 2:45	Discussion Leader: **Dr. L. LaPadula**, *The Mitre Corporation*
2:45 - 3:05	*New Paradigms for High Assurance Software* **Dr. J. McLean**, *Naval Research Laboratory*
3:05 - 3:25	Discussion Leader: **E. Leighninger**, *Dynamics Research Corporation*
3:25 - 3:45	*BREAK*
3:45 - 4:05	*Managing Complexity in Secure Networks* **Dr. D. Bailey**, *Galaxy Systems*

4:05 - 4:25 Discussion Leader: **Dr. M. Abrams**, *The Mitre Corporation*

4:25 - 4:45 "Best Paper of the New Security Paradigms Workshop"

4:45 - 4:55 Discussion Leader: **E. Leighninger**, *Dynamics Research Corporation*

4:55 - 5:30 Panel Discussion
Dr. J. Dobson, *Newcastle upon Tyne*
Dr. D. Bailey, *Galaxy Systems*
Dr. D. Denning, *Georgetown University*
H. Hosmer, *Data Security, Inc.*
Dr. L. LaPadula, *The Mitre Corporation*
Dr. J. McLean, *Naval Research Laboratory*

Wednesday October 14, 1992

9:00 - 9:05 *Introduction*
Dr. J. Dobson, *Newcastle upon Tyne*

9:05 - 9:25 *The Multipolicy Paradigm*
H. Hosmer, *Data Security, Inc.*

9:25 - 9:45 Discussion Leader: **Dr. T. Haigh**, *Secure Computing Corporation*

9:45 - 10:05 *Metapolicies II*
H. Hosmer, *Data Security, Inc.*

10:05 - 10:25 Discussion Leader: **Dr. L. LaPadula**, *The Mitre Corporation*

10:25 - 10:40 *BREAK*

10:40 - 11:00 *Separation Machines*
Dr. J. Graff, *Amdahl*

11:00 - 11:20 Discussion Leader: **M. Smith**, *AT&T*

11:20 - 11:40 *Mediation and Separation in Contemporary Information Technology Systems*
J. Heaney, *The Mitre Corporation*

11:40 - 11:55 Discussion Leader: **E. Leighninger**, *Dynamics Research Corporation*

11:55 - 12:00 *Wrap Up*, **Dr. J. Dobson**, *Newcastle upon Tyne*

Presented at the ACM SIGSAC New
Security Paradigms Workshop, Little
Compton, Rhode Island. September 1992.

Managing Complexity in Secure Networks

D. Bailey,
Galaxy Computer Services
Santa Fe, NM

July 1992

Abstract

The "Security Island" physical security paradigm, on which we base our concepts of protecting computer systems, derives from notions of centralized control and isolation. Implicit in this view is the need for "global understanding" of the system being protected: it must, in principal, be possible for a single person to know about all of the data paths and security controls within the system. Otherwise, it is not possible to analyze adequately the protection afforded by the system. As a system grows in size and complexity, maintaining global understanding becomes increasingly difficult, and ultimately it is impossible. This note suggests two alternate paradigms that show promise of surviving the complexity threshold at which the security island paradigm collapses. The "Secure Telephone" paradigm distributes protection responsibility to data objects. The effect of this is a dramatic limitation on the number of network components that are security relevant and a corresponding reduction in the complexity of the security problem. The "VIP Protection" paradigm uses graded protection and assurance to protect sensitive resources. Using strong protection only where it is needed and weaker measures in less sensitive areas reduces system cost and focuses security attention where it is most needed.

Do We Need a New Paradigm?

A large or complex computer network poses many difficult security problems. The hardest of these seem to stem directly from the complexity of the network. The trouble is that our ideas of how to protect computer systems and networks* are derived directly from our past experiences with physical security systems. The physical security paradigm, which we might call "security islands," is based on isolation and hierarchical control. As network size and complexity grow, hierarchical control becomes increasingly difficult to manage. It is no longer possible to "understand" what the system really does, it is no longer possible to analyze the implications of change, and it is no longer possible to decide that development is being performed correctly.

This paper will explore the security islands paradigm and where it leads, followed by two "new" paradigms (neither of which is at all new). One paradigm is based on rejecting hierarchical control, and the other is based on rejecting isolation.

*For the purposes of this paper, any distinction between the ideas of "computer system" and "computer network" is irrelevant. In the remainder of the paper I will use the terms "system" and "network" interchangeably.

Security Islands

The security island paradigm is the primary paradigm used for designing physical security systems for fixed plant sites. A sensitive operation is located in a building somewhere. To protect it, we establish a security perimeter, build a fence, put guards at the gates, and control who can enter and exit.

Variants on this scheme are used when the requirements vary. Sometimes the perimeter is not made obvious with a fence and the guards are less visible. Sometimes the perimeter is arranged so that the public can enter part of the facility. Sometimes it is necessary to segregate part of the site population from other parts. The basic design, however, remains the same. One person is in overall control of facility security. That person knows what assets are being protected and how the protection is being accomplished. He or she is in a position to analyze the effects of changes and to establish whether security changes have been made correctly when the operation changes.

When it first became necessary to protect computers, the job was assigned to the physical security manager for whom the techniques to be applied were obvious. In the early days, when the system was a single machine that ate cards and produced listings or tapes, the techniques applied by the physical security manager worked very well. Over the first decade of computer security experience, a large body of knowledge was accumulated and became well entrenched. The techniques that developed had only one minor problem—they were inadequate for the remote access time-sharing systems that were coming into vogue at the end of the period.

Now, another fifteen years later, we are left with a legacy of physical security attitudes and practices that have been gradually bent and stretched to their limits to accommodate new technology. Computer networks have grown large despite the active resistance of security managers. During this growth, we have retained the attitude that a network is a collection of separately accredited components that have to share data.

For a closed system of two computers, it is relatively easy to decide what controls are needed. As this system expands into a larger network, it becomes necessary to compute the transitive closure of every allowed flow in order to make network-high or global statements about the protection of data. This cannot be done without computational assistance; one's intuition no longer works because the flows may be complicated. This assistance must be obtained from the very systems that the security manager is loath to trust. It is not difficult to see why the security of networks is thought to be a hard problem. It is also not difficult to see why security managers typically believe that the computing people are out of control; and computing people, for whom new network connections are easy, think of security managers as paranoid bureaucrats.

We should not be terribly surprised at these results. The separate accreditation of network components is the network analog of creating a new security area at the protected site (another security island). Adding several thousand new physical security areas at the site would make the site unmanageable—and it doesn't work any better with computers. It is attractive to accredit separately because the other alternative, that of re-evaluating the entire network every time a new node is added, is an obvious failure. Unfortunately, it seems to be necessary for a single person to understand the operation of the entire network

in order to understand its security properties. For many existing networks, it is already impossible for a single person to understand the entire network operation. Clearly, any network can grow to this state. For example, the Department of Energy presently has at least three networks of approximately 10,000 accreditable nodes. This is already too big. By the end of this century, DOE will have responsibility for at least one network with approximately 100,000 accreditable nodes. This is far too big.

As a result of the size of many existing networks, any security policy or management scheme based on "global understanding" of the network is bankrupt. Such a policy, while adequate for a small network will ultimately be insufficient. The result of long term development may be failure of the network to provide adequate service to its customers because its security managers or accreditors are conservative. Development can also be continued beyond the point where it is adequately secure because the developers are persuasive. Most likely, both results will occur.

The natural human response to this situation is to modularize. We want to break the network into independent pieces that can be understood separately and whose interactions can be analyzed pairwise. Within the Department of Energy community, we have used a concept called partitioning to provide the needed modularity. A **partition*** is a division of the components of a network for access control purposes such that no component is in more than one partition. The systems within a partition that serve users directly must have the same protection requirements, and the users must satisfy a common clearance requirement. Relatively free exchange of information is allowed within a partition. For the purposes of security analysis, the systems within a partition are all equivalent. One can think of them as a single system, even though they may not offer this functionality to their users.

Partitioning, applied to collections of operating systems, was an adequate paradigm for the '70s and the early '80s when interactions were between separate systems and were fairly simple. It put off the inevitable for another decade. It is much less adequate today when intersystem interactions are more numerous, occur at a lower level of detail in the network, and are less obvious to the user. Network File Systems (NFS) and diskless workstations, where the use of the network to obtain a requested piece of information is completely hidden from the user, are good examples of this new style of interaction.

The new types of system interaction demand a new paradigm for modularizing security, while also making modularization more important. Since the interactions are occurring (that is, are initiated) at a lower level in the system, the modularization must be finer grained than partitions. Instead of discussing how this VAX communicates with that IBM system, we must move down to the level of interactions between processes.

The following discussion offers two new paradigms for modularizing network protection. In the first, we will categorize systems based on the amount of internal mechanism needed to provide adequate security in the operating environment seen by the system. Systems requiring the same degree of trust will be characterized by an index number called the "Trust Index." The idea is to focus attention and to place security mechanisms, which are expensive, where they are most needed. The other paradigm, based on ideas of the secure telephone system, reflects the need for centralized control of the network. Security

*Boldface words are being defined. The definition follows immediately.

responsibility is distributed to its logical extreme to see what the effects might be. Neither one of these paradigms solves all of the problems or renders security easy. Security is not easy and many hard problems still have to be solved. However, it may be possible to survive the imminent collapse of our ability to protect data in large networks.

Protecting VIPs

The problem of protecting important people (a physical security problem), offers some interesting insights. To begin, the problem is dramatically different from the problem of protecting fixed plants. There is no fixed or well determined security perimeter. The asset is continually moving through an environment that is largely friendly but is presumed to contain some very hostile elements. It may or may not be possible to identify these hostile elements with their unknown intentions.

This problem seems more difficult than protecting a fixed and slowly changing computer network, but it is solved every day. We should be able to draw some lessons from how it is done which can be applied to the network problem. One technique that is relatively easy to transfer is the idea of many layers of protection. Protection of a Head of State includes at least four layers. The strongest protection is immediately around the VIP, and the layers get progressively weaker as the distance from the asset increases. The outer layers are not only weaker, however, they are also less trusted than the inner layers. The outermost security layer for a Head of State consists of increased surveillance by the local police. While the local police perform a valuable service, they are completely untrusted by the VIP's immediate bodyguard. This idea of layers of protection transfers very readily to various levels of sensitive data.

Sensitive data requires protection. More sensitive data requires more protection. However, sensitivity alone does not mean that protection mechanisms must be built into the system. Often, better protection of data can be obtained by physical means. The requirement for an internal protection mechanism arises from the need to operate over a range of sensitivities either in the data, in the authorization of the users, or both. Because of the range of sensitivities, we must trust the system to make critical decisions: should this person obtain that data; should this process perform that function. "More sensitive data requires more protection," may simply mean a stronger lock on the door. To reiterate, a range of sensitivites or authorizations imposes a need for internal mechanism. A wider range imposes a need for stronger mechanisms. For example, let's look at a system which processes Secret data for a community of users all cleared at that level. The internal mechanism becomes one of convenience to allow the users to do their work without getting in each other's way. Another system which processes Top Secret data for a community of users all cleared at the Top Secret level would have the same internal mechanism to do this job, but the physical security might need to be different. A system which had both Secret and Top Secret data would require a stronger internal mechanism in order to keep the two types of data separate. This system, then, would have an internal security boundary which the system would have to manage in order to keep the two types of data separate.

Different components of a network see different local environments. Each component

lives in some neighborhood of the network that consists of network components with which it can directly communicate. For example, a network neighborhood might consist of components on a local area network. Components that only communicate with each other indirectly would be in different neighborhoods. Network components that live in different neighborhoods may have different levels of sensitivity. If the network as a whole processes a wide range of sensitivities, then some components will be faced with a range of sensitivities and will require enough mechanism and enough trust (assurance of correctness) to handle that range of sensitivities. However, as with the bodyguard, there is no reason to suppose that all components of a network need to be trusted to the same extent. Hence, the Trust Index, a label used to distinguish components that must be highly trusted from those that can be trusted less, becomes a useful concept.

Trust Neighborhoods satisfy the need to modularize. Using them, one can divide the network into regions or neighborhoods that are equivalent in the sense that all connected components in the same neighborhood "see" the same protection environment and process the same range of sensitivities. It is then possible to consider each neighborhood as a unit and assess the requirements for controlling flow between the units.

As with partitioning, the trust neighborhood decomposition imposes an equivalence class structure on the network where the number of equivalence classes is much smaller than the number of components. Even better, the number of classes does not usually grow when new nodes are added to the network because they are usually added in existing neighborhoods. This eases understanding of network security and provides a way out of the growth problems described earlier.

A formal data flow policy can be described that provides rules for moving data between different trust neighborhoods. One can limit the exposure of data by forbidding direct communication between components that differ greatly in trustworthiness. Instead, data flows gradually from highly trusted components through components that require less trust because they are protecting less and making less complicated decisions. For example, the policy would not allow direct connection of an "open"* workstation to a system processing sensitive data, but it would allow indirect, appropriately protected data flows. A somewhat analogous situation can be seen in VIP protection. Generally, people are allowed to move somewhat freely into and out of the immediate area occupied by the asset. However, high speed movement that appears to be directly toward the asset would be stopped early and as far away from the asset as possible, which illustrates a soft, permeable boundary that stiffens rapidly as a function of the rate of penetration.

The Secure Telephone System

There are several ways to simplify the security structure of networks. The Trust Neighborhood model described in the previous section suggested a decomposition of network components that focuses design attention on those components that need internal security mechanisms to fulfill their responsibilities. This model may be very useful in connection-

*An open system means that the system can be made available to users without consideration of their clearance. It does not necessarily mean that use is completely unrestricted.

rich environments where it is difficult to establish the security perimeter or where external connections are needed even though sensitive data is processed locally. The trust neighborhood decomposition rejects the notion that complete isolation is necessary to be able to protect sensitive data. The following paradigm rejects the notion that hierarchical control is necessary.

The secure telephone system consists of telephone units called STUs (secure telephone unit) and a centralized key distribution system (KDC) for creating encryption keys. The STUs have two main functions: (1) to provide secure voice communication with other STUs and (2) to recognize and refuse to communicate with STUs that have been lost or stolen. They guarantee a level of communication security by negotiating with the remote STU and with the key distribution center at the time a call is established. All of this negotiation is hidden from the users. The user simply pushes a button labeled SECURE and waits to be told if he may continue the call at some level of security.

In the secure telephone system, security responsibility is distributed outward to the users. A network mechanism is provided that obviates any requirement for security mechanisms built into the network itself. It is not even necessary to know that secure communication is possible at the time a call is initiated. The connection is established, the need for and possibility of security is negotiated, and the secure connection is established using a trusted third party. After this occurs, the communicating parties still have the option of deciding not to communicate—the final access control occurs using the secure connection.

Security in the telephone system is established using a mixture of a particular encryption technology and human judgment. A computing network could be established today using this technology, but it would not be a very capable or convenient network. A more effective but more radical approach would be to distribute responsibility for protecting data to the data itself.

Suppose that an object* were really able to guarantee that the only way to get to its data is through its methods.† Let's consider the potential effects on the security requirements for a network. To be gelatinous, if not concrete, consider a local area network comprised of workstations (not necessarily single user, but probably one at a time except for NFS mounts), and a print server. The OOP paradigm works in the following manner. A process object obtains information from a data object by sending it a message. Access control is performed by the data object according to the object's policy. The object's policy may be different from every other object's policy and may be quite complex. It may, for example, include consideration of user identity, clearance, and role. It may also include consideration of the local processing environments of both the requesting object and itself, as well as other information.

If protection responsibility and capability are given to a data object, then many other requirements could disappear. There would no longer be any security requirement on the transmission medium itself. There would be no security requirements on a file storage system other than being able to return objects that had been previously stored. This would take care of two currently pressing issues: how to protect data when all the users use removable media, and how to securely implement Network File Storage systems. Even the

*that is not object as in subject and object, but object à la OOP (Object Oriented Programming)

†Encryption may be a way to do this, but it may not be the only way or the best way.

access control requirements for workstations would disappear—users would establish access rights directly with the data object, not the underlying system.

Operation of a print server would be somewhat different than it currently is. The print server would need to establish dynamically that the data to be printed is printable there. Security restrictions might prevent creating particular documents on particular printers, and these restrictions could vary in time. For example, what is permitted now may not be allowed in ten minutes if a particular person leaves the room. Thus, when requested to print, a data object would not respond with a stream of text that could be sent to a printer. Instead, it would respond by creating a printable object that would be able to decide, through its methods, whether to print on a particular printer. This object would be sent to the print server, would negotiate with the print server the conditions of printing, would print the requested number of copies, and then self destruct.

Obviously, there are many difficult problems that need to be solved to make this vision a reality. The attractiveness of the paradigm comes from its ability to localize and simplify network security concerns. This happens partly because the need for hierarchical administration and global understanding of the network have been eliminated. An object containing sensitive data can be moved freely around the network. It is no longer necessary to consider whether the object can be accessed in a particular location before sending it there. In the secure telephone system, the central administration does not have a precise idea of how big the network is or where all the nodes are located. Likewise, the central network administration need not know the extent of the computer network or exactly where sensitive data is processed. The central administration establishes rules for local protection and local connection to the larger network, implements most of them in the object support mechanism, and leaves the rest to local administration.

Conclusions

This paper makes the argument that centralization, isolation, and hierarchical control will ultimately defeat our ability to make large and complex networks that we can trust. In fact, this has already happened. It suggests two new ways of looking at network security that reject traditional notions of network security management and facilitate a movement to more complex, more capable networks that can be trusted. Although not argued here, both paradigms are formalizable and, hence, it will be possible to study them systematically and demonstrate systematically that implementations have been made correctly.

Presented at New Security Paradigms Workshop, ACM SIGSAC, Little Compton, RI, Sept. 22-24, 1992.

A New Paradigm for Trusted Systems

Dorothy E. Denning

Georgetown University
Computer Science Department
Washington, DC 200057
202-687-5703
denning@cs.georgetown.edu

The Current Paradigm and Breakdown

The current paradigm for trusted computer systems holds that trust is a property of a system. It is a property that can be formally modeled, specified, and verified. It can be "designed into" a system using a rigorous design methodology. For high levels of assurance, the design methodology uses formal models and methods in order to "prove" that trust is present.

This paradigm underlies "The Department of Defense Trusted Computer System Evaluation Criteria," [3] commonly called the "Orange Book," and its companion "rainbow series" reports. In this paper, I will refer to these documents as the "Criteria." The Criteria specifies a methodology for modeling, designing, and implementing a system that builds trust into a system, and a process for proving to an evaluator that the methodology has been followed. For a description of the Criteria and the evaluation process, see Chokhani [1].

Application of the Criteria has been fraught with problems for both developers and evaluators. Steve Lipner clearly articulated this breakdown in the keynote address at IFIP-SEC 91 [4]. The problems he identified include:

1. Systems are not operated in their evaluated configuration. Evaluated systems are penetrated because they are not properly configured or operated.

2. The Criteria apply to operating systems products, whereas actual operating environments include heterogeneous networks and applications.

3. Applications must run with "privilege," overriding the operating systems controls. Evaluation becomes irrelevant. There is no experiential basis on which to build application-level criteria.

4. Real systems are vastly more complex than their security models. The vendors learn what system settings, tools, and documentation are needed from the experiences of their customers with their products.

5. The security management documents are thick and there is a forest of controls. The paperwork

required of vendors is an enormous burden.

6. By the time a product has been evaluated, it is obsolete.

7. The Rating Maintenance Program (RAMP), which was designed to allow vendors to self-evaluate new versions of a product, imposes a plethora of paperwork, checking, bureaucracy, and mistrust on vendors.

8. No one knows what a class C2 system is. Part of the problem lies in applying an abstract model of subjects and objects to real systems when it is not at all obvious what should be subjects and objects in the system. Because of these problems, it has been necessary to produce "interpretations" of the Criteria. The interpretations grow and change as new systems are evaluated, but nonetheless remain ambiguous.

Lipner offers some suggestions for improving the process. While his suggestions are likely to alleviate some of the problems, I propose that we also rethink the question "What is a trusted system?" My initial investigation into this question suggests that the current paradigm, which treats trust as a property, is inconsistent with the way trust works in the world. By shifting to a paradigm that is consistent with the realities of trust, we may be able to produce trusted systems at considerably reduced cost, effort, and aggravation. I shall propose such a paradigm here, and I invite the reader to explore its implications with me.

The need for a paradigm shift is not limited to the domain of security. Peter Denning [2] has noted that software quality is held as a property that can be designed into a system by a four-stage process: formulate the requirements, develop formal specifications for the requirements, develop programs from the specifications, and demonstrate that the programs meet the specifications. He proposes a shift in paradigms by reframing the question "What is software quality?" to "How do we satisfy the customers of our software?"

The paradigm for trusted systems presented here similarly focuses on producing systems that satisfy customers, in this case, systems that customers trust in the domain of security.

What is Trust?

Trust is an Assessment

The word "trust" is used with people, organizations, and objects. It is an assessment that a person, organization, or object can be counted on to perform according to a given set of standards in some domain of action. As an assessment, it is a declaration made by an observer rather than an inherent property of the person, organization, or object observed.

For example, we may trust a person to speak truthfully, keep promises, arrive on time, give an entertaining talk at a conference, represent our concerns at an important meeting, lead a project, implement a program, fly an airplane, or perform open heart surgery. We may trust an organization to keep our records confidential, deliver certain types of products or services, or refund our money if we are unsatisfied. We may trust an airplane to not crash, a bridge to not collapse, the groceries we purchase to not be contaminated or poisonous, or a program to perform its stated function and not have undesirable side effects.

An assessment of trust is always relative to a domain of action. We may trust a person to give a stimulating lecture on computer crime, but not trust them to fly an airplane or cook a Thai dinner. We may trust a woodworker to produce a cabinet of exceptional quality, but not trust them to deliver it on time. Thus, people are not simply trusted or not trusted, but rather trusted or not trusted in a particular domain. However, we often lose the distinction of domain, generalizing assessments of trust across domains. For this reason, we often hear people say things like "This person cannot be trusted."

Likewise, an assessment of trust is always made against a set of standards in the domain of action. These standards evolve in communities of people who interact and coordinate action together, and they may differ from one culture to the next. They are often loosely defined or subjective, for example, standards for a "good teacher," a "good restaurant," or a "good department." They may be so ingrained in our culture that we are not even consciously aware of their presence. Yet they play a critical role in our coordinated actions in the world.

The domains and standards for trust change over time as new technologies come to market and new breakdowns occur. A few years ago, nobody was concerned about whether a floppy disk might contain a computer virus or other form of malicious code. Now people are reluctant to trust a disk if they are not sure of its origin. The Tylenol scare led to higher standards for packaging drugs and other goods.

An assessment of trust may or may not be grounded. It is grounded if evidence can be produced that the standards are met. Otherwise it is ungrounded. In many situations, it is less important whether an assessment is grounded than whether it is believed. People act out of their beliefs even when there is no evidence to support them.

How Assessments of Trust are Made

We make assessments of trust based on our experiences in the world. As we interact with other people, organizations, and objects, we observe the effects and form our assessments. If a person consistently keeps their promises, then we trust that person to keep future promises. But if they fail to keep a promise, we may begin to distrust them. Similarly, if we try a new restaurant and have a good experience, then we may make an assessment that the restaurant is excellent. However, if we go back and have a bad experience, we will change our assessment and possibly never return. We often make assessments of trust based on a single incident; this is why first impressions are so important.

If we do not have direct experience with a person, organization, or object, we will make an assessment of trust based on the declarations of others whom we trust. If a person whom we trust says that another person is an entertaining and stimulating speaker, then we may accept their assessment and invite the person to give a talk at a conference. If a restaurant critic or friend reports on a new restaurant, then we may use their assessment to determine whether to try the restaurant. If a popular computing magazine reports that a particular vendor provides better service than a competitor, we may decide to order products from that vendor. We make purchases, hiring decisions, travel plans, invitations, and other decisions based on what others say when our own experience is inadequate.

There has been a growing industry relating to the buying and selling of assessments of trust. This industry includes organizations such as Consumer Reports; consultants and consulting firms with expertise in specialized domains; and magazines, newsletters, and articles which evaluate products, services, and organizations. Although we often rely on the assessments of others, we give greater

weight to our own experiences, and we will not accept another person's assessment if it contradicts our own experience. Instead, we may lose trust in the other person's assessments. We are most influenced when we lack experience of our own.

We thus ground our assessments of trust on our personal experiences and on the experiences of others whom we trust. We seldom base our assessments on mathematical theories. The Golden Gate Bridge is trusted, not because someone proved mathematically that it would not collapse, but rather because it has withstood over 50 years of service. In 1987, it passed an impromptu "proof test" by supporting the largest load ever, 250,000 people. By comparison, the Tocoma Narrows Bridge, which was built using the same theory, was destroyed by wind in 1940 [5,6].

This does not mean that formalism has no role in the establishment of trusted products. Formal theories and methods may be used to validate certain aspects of a product, e.g., to show that a circuit design or software module will satisfy certain properties. These methods can help the developers establish trust in their product before it is released. However, the product itself will be assessed by users according to their standards. If a software product shows no evidence of containing malicious code after several years of use, then it will be trusted to be non-malicious regardless of whether that property was formally proved.

Trust is a Critical Element of Markets

Assessments of trust are thus formed and shared in a world where we interact with the people, organizations, and objects around us. This world is also a marketplace of transactions, and the value of a person, organization, or object in the market will be determined to a large part by the amount of trust that others have in them. If person has a reputation of being a highly talented athlete and of high integrity, then that person will have many opportunities in the market. Similarly, if a service provider has a reputation of providing exceptional service at competitive prices, then it is likely to do well. But reputations are volatile. Once a person or organization acquires a reputation of being untrustworthy, it can be hard to overcome that reputation even if the assessment was poorly grounded.

The word "market" is being used in a very loose way to refer to the space of all transactions, including social transactions that do not involve money. A transaction is any exchange between two parties. The transaction may involve loaning a book in exchange for the right to borrow one in the future or even for the friendship that will follow from the loan. A conversation can be regarded as an exchange where two people share information, beliefs, thoughts, and emotions.

In this market, people can trade as they choose, subject only to their own ability to make offers that are desired by others, and by the regulations and rules that are imposed by governments and private organizations. The viability of a person, company, or product in the world is strongly determined by the trust they evoke in those they wish to interact and trade with. The market will eventually weed out people, organizations, and products that are considered untrustworthy, though this may take time if there is little or no competition in that domain. In a sense, the market determines the criteria for trust based on the needs and demands of the people.

In the domain of aircraft, for example, the market has demanded planes that do not crash. If a plane crashes and the cause of the crash can be attributable to a design flaw, then people will not fly on planes of that type. This happened to the DC-10 after one incident, and there are people who still avoid it.

The New Paradigm

The current paradigm of treating trust as a property is inconsistent with the way trust is actually established in the world. It is not a property, but rather an assessment that is based on experience and shared through networks of people in the world-wide market. It is a declaration made by an observer rather than a property of the observed.

In the new paradigm, we see that a "trusted system" is one that produces assessments of trust. These assessments are based on standards of performance and are grounded in observable behavior of the product in the marketplace. The standards for trust will change as new technology, new threats, and new practices are introduced in the market. Moreover, the assessments about a particular system will be continually remade each time the system is used. Ultimately, a system is trusted if and only if its users trust it.

The new paradigm has several implications relating to the Criteria and to producing trusted systems. The following touches briefly on these implications. Further study is needed to develop a more complete understanding of the proposed paradigm shift.

Security Criteria

At first glance, it might appear that the current Criteria recognizes that trust is an assessment rather than a property since the security rating assigned to a system (C2, B1, etc.) is an assessment. However, the Criteria are based on the assumption that trust is a property that can be built into a system following specified design methodologies rather than the premise that trust itself is an assessment made by users based on how well the observed behavior of the system meets their own standards.

In the new paradigm, security criteria would articulate the (possibly unstated) standards that users employ when making assessments of trust; that is, they would formulate the concept of customer satisfaction in the domain of security. They would emphasize those features that customers are most concerned about, for example, protection against break-ins and viruses, simple access controls, ease of use, and product support.

Since users do not particularly care how a system is structured internally or the methodologies used during development, the security criteria would not specify how a system should be modeled, structured, designed, or developed as in the current Criteria. For example, there would be no concept of security kernel, trusted computing base (TCB), or formal security policy model. There would be no requirements on system architecture, design specification and verification, or configuration management.

They standards would be specific to different types of products and stated in terms of actual users, processes, and entities rather than abstract subjects and objects. Thus, they would not require "interpretation" of an abstract security model and they would be readily understandable to users and developers alike.

To illustrate, the standards for operating systems might include discretionary access at the level of individual files and users, logging of all successful and failed login attempts, and break-in prevention.

The standards for database systems might include discretionary access at the level of records, attributes, and individual users, and logging of all database accesses at the relation level and all updates at the record level. The standards for virus protection software might include the ability to detect any virus in a specified list and the ability to remove any detected virus. The standards for networks and communication systems might include optional encryption using the Data Encryption Standard.

There may be a common set of standards applicable to all types of products, for example, standards for product service and support. Since many security problems arise from improper installation or operation, or from flaws that are discovered after the product has been released, product support is a significant factor is customer satisfaction and assessments of trust.

The standards might be classified according to whether they are required for a certain "level of security" or for certain types of environments (banking, hospital systems, etc.). For example, being able to withstand penetration attacks from legitimate users might be associated with a higher level of trust than preventing break-ins. A product could be evaluated by checking off the standards that it meets.

"Security benchmarks" could be included with some of the standards. For example, consider a standard for break-in prevention. This standard could be assessed through a "break-in benchmark" that could be run against a system to see if it succumbs to certain attacks, for example those that use password cracking programs or exploit potential network protocol vulnerabilities. One can envisage other benchmarks, for example, to assess the ability of a virus protection package to detect viruses.

This approach of assessing observable behavior and of using benchmarks is not new. Indeed, it has arisen naturally in the market in response to customer needs. There have been many published articles that rate or compare security packages in concrete terms, and vendors and researchers have developed software tools that can test for the presence of weak passwords, improper defaults and system settings, and various other vulnerabilities. All of these assessments and tools have been developed with the goal of meeting the needs of customers, and are entirely consistent with the way trust works in the world. Thus, the paradigm described in this paper is already practiced in the commercial world, and the existing practices provide a useful starting point for determining security criteria.

Although the Criteria is based on a model of trust that is inconsistent with the way trust works, it offers much towards the construction of new security criteria. Many of the requirements relate to functionality needed by users, and while many are abstract, they could be made concrete. The requirements for penetration and covert channel testing identify areas where benchmarks could be created, although it is unclear that protecting against most covert channels corresponds to any real-world market need. The security criteria would be driven by market forces. They would reflect the current standards for trust in the market, and they would change with market needs. They would be developed by or at least with users representing a variety of different customer bases.

Although there could be more than one set of standards, a national or international standard has the advantage of providing industry with a clear set of guidelines. The standard(s) could be produced by the government through the current NIST/NSA effort or by other standards groups, for example, ANSI, the IEEE, and ISO.

Although security criteria articulate community standards for trust, a system that meets the criteria is not necessarily trusted. Ultimately, trust is always determined by users whose needs may deviate from the community standards. This underscores the importance of product support from a vendor.

Producing Trusted Systems

In the new paradigm, vendors would be free to design and develop systems using any architecture and methodology they choose. The security criteria would not impose any particular structure or methodology on the customers. Security kernels, formal models and methods, and other developmental requirements in the current Criteria would be used only to the extent that vendors perceive that the return on their investment justifies the cost. The requirements in the current Criteria, coupled with the costly evaluation process, have led many vendors to conclude that it is simply not worth the effort to develop systems at those levels where formal methods are required. Removing these requirements opens up the possibility of considerable innovation in the development of trusted systems. Researchers may be able to uncover structures and methodologies that produce trusted systems at considerably reduced cost.

The current Criteria were developed with the objective of eliminating all security risks, at least at the higher levels. By adapting a particular architecture and following a specified design methodology based on formal specifications and proofs, security risks would be avoided. This risk-avoidance strategy has the disadvantage of inhibiting innovation and progress in system architecture and development. If followed to its extreme, it will guarantee that "trusted systems" are archaic and not cost-effective. As illustrated by Petroski [6], progress in engineering comes only when designers take risks. Taking risks is essential in order to build systems that are more economical, functional, or aesthetically pleasing than their predecessors. Moreover, we learn more from our failures than our successes, and progress depends on failures. A strategy of creating criteria that eliminate security risks is especially dangerous because we lack worked examples, especially for applications such at database systems, transaction processing sysems, and heterogenous networks. A better strategy is to encourage risk taking while disseminating knowledge about failures through channels such as CERT and security publications.

Summary

The current paradigm for trusted systems holds that trust is a property of a system. I have argued that this paradigm, which underlies the Criteria for trusted systems, is inconsistent with the way trust works in the world.

I then examined the concept of trust, showing that trust is an assessment made by an observer about a person, organization, or object observed. These assessments are formed and shared in a world-wide market where people interact with each other, with organizations, and with objects. Our own assessments are based on our personal experiences and on the assessments of others whom we trust.

This understanding of trust as an assessment formed in a market leads to a radically different approach to the development of security criteria. In this paradigm, the criteria would be a set of standards directly related to customer satisfaction. The standards would reflect current market requirements, be specific to different types of products, and be stated in terms of actual users, processes, and entities rather than abstractions such as subjects and objects. They would continually evolve to respond to new technologies, new threats, and new demands in the market.

The criteria would not impose requirements on the internal structure of a system or on development methodologies. The vendors would be free to choose their own methods for producing secure

systems. Their systems will be evaluated according to market-based criteria for customer satisfaction, and they will be trusted as long as they meet the evolving standards and needs of the customers.

Further study is needed to determine whether the proposed approach is sound for at least commercial systems if not military ones. If it is, then additional work is needed to identify the current community standards in order to formulate new criteria. Beyond that, the approach opens up the possibility of new security architectures and methodologies, and of news products that support the evaluation process, in particular security benchmarks.

Acknowledgments

I am grateful to Peter Denning, Hilary Hosmer, Bob Lawton, and Steve Lipner for their generous and helpful comments on an earlier version. Once again, my colleagues saved me from at least one round of public embarrassment.

References

1. Chokhani, S., "Trusted Products Evaluation," *Comm. of the ACM*, Vol. 35, No. 7, July 1992, pp. 64-76.

2. Denning, P. J., "What is Software Quality?" *Comm. of the ACM*, Vol. 35, No. 1, Jan. 1992, pp. 13-15.

3. Department of Defense, "Trusted Computer System Evaluation Criteria," DOD 5200.28-STD, December 1985.

4. Lipner, Steven B., "Criteria, Evaluation, and the International Environment: Where Have We Been, Where Are We Going?" Proc. IFIP-SEC '91; also in RISKS-FORUM Digest 12.46, October 1991.

5. Petroski, H., "Making Sure," *American Scientist*, Vol. 80, March-April 1992, pp. 121-124.

6. Petroski, H., *To Engineer is Human, The Role of Failure in Successful Design*, Vintage Books, 1992.

Executive Summary

Perspectives and Progress on International Criteria

Eugene Troy, Co-Chair
National Institute of Standards and Technology
Ron Ross, Co-Chair
National Security Agency

David Ferraiolo, National Institute of Standard and Technology
Eugene Bacic, Canadian System Security Centre
Jonathan Wood, UK Department of Trade and Industry

Through the experiences, efforts, and cooperation of various national and international bodies, an evolutionary and spiralling process of Trusted Information Technology (IT) product and system evaluation has emerged. This process has included the efforts of Germany, France, Great Britain, the Netherlands, the European Community, Canada, and the United States. Starting with the Trusted Computer System Evaluation Criteria (TCSEC) and its associated evaluation process, a number of criteria and evaluation approaches have and will continue to be developed.

It is universally recognized that government and commercial institutions rely heavily on information processing systems to meet their operational, financial, and information requirements. The integrity, availability, and confidentiality of key software systems, databases, and data networks is a major concern in all industrialized nations.

The TCSEC was the first publicly available document that expressed general security requirements that could apply to a specific class of technology, e.g., operating systems. The TCSEC was originally published in 1983 and revised in 1985. It represented the culmination of many years of effort to address IT security issues within the Department of Defense (DoD) classified world. Since its publication, the TCSEC has influenced vendors, consumers, and the authors of other requirements documents both in the US and abroad.

The TCSEC was developed to meet several objectives. First, to serve as a "metric" to measure the amount of security present in a computer system to be used for the processing of classified or sensitive information. Second, to provide guidance to the developers as to what security features to build into their planned systems. And third, to provide a method for uniformly specifying security requirements in acquisition specifications.

The TCSEC is divided into four hierarchical divisions. The divisions are further divided into numbered hierarchical classes (e.g., C1, C2) with higher numbered classes providing greater degrees of security. Each class represents the overall level of trust placed in a system, specifying a collection of requirements

in the form of features and assurances. A given class includes all the features and assurances of the previous class along with additional, more stringent features and assurances.

With the advent of the proposed European Community as a political and economic force, a more coordinated method of defining computer security standards was needed. Germany had the ZSIEC, France had the "Blue-White-Red Book," Great Britain had the "Green Book," and the United States had the "Orange Book." Four European counties (Germany, France, Great Britain, the Netherlands) combined their knowledge to created a harmonized security criteria referred to as the Information Technology Security Evaluation Criteria (ITSEC). While harmonizing these criteria, consideration was taken to achieve commonality among their own countries as well as with the United States. For this reason, the members considered the TCSEC and elected to expand many of the premises while adding additional criteria and more detail. Version 1 was published in June of 1990, with a second version released June 28, 1991. Currently, the ITSEC is fixed for a two-year provisional period before further consideration is made to its modification and update.

The ITSEC provides a basis for evaluating any specified set of IT security functionality in terms of correctness and effectiveness. It provides a methodology for gaining confidence in the correctness and effectiveness of security functions implemented in IT products and systems by use of a set of well-defined assurance evaluation levels.

The ITSEC was the first criteria to consider separation of security features and security assurances. The ITSEC does not specify security functionality requirements but permits the definition and use of a variety of functionality profiles. The ITSEC describes an approach called a Security Target for specifying and justifying the security functionality and level of assurance required in a particular product or system. The ITSEC prescribes a general approach that can be used to evaluate any combination of functionality that a vendor or sponsor sees fit, something the TCSEC levels do not provide.

The Canadian System Security Centre (CSSC) of the Communications Security Establishment (CSE) recently published the final draft of the Canadian Trusted Computer Product Evaluation Criteria (TCPEC). The TCPEC provides many of the same benefits of the ITSEC by being adaptable to a variety of IT-based products and applications. It takes the approach of separating features and assurance. However, the TCPEC goes beyond the ITSEC in delineating both security feature requirements and assurance requirements. As such, this criteria serves two purposes: first, it provides a metric for evaluating trust placed in computer products; and second, provides a guide to manufacturers as to what security features to build into their products. The TCPEC allows for flexibility in choosing appropriate functionality based on environmental needs in terms of functional building blocks.

In the United States, a project has been underway since November 1991, to develop the US IT Security Standard, sometimes called the "Federal Criteria." This project has the express goal of revising the existing TCSEC by taking into account all relevant later work. That work especially includes the ITSEC and the TCPEC. The Federal Criteria is being developed jointly by the National Institute of Standards and Technology (NIST) and National Security Agency (NSA) to provide a comprehensive criteria for specifying, developing and evaluating IT security.

The Federal Criteria will include a more responsive means of specifying applicable security needs than the TCSEC. To achieve this goal, it will: introduce a new form of requirements specification to provide a clearer presentation of needs; elaborate upon the TCPEC's concept of functional building blocks; refine the ITSEC concept of a "Security Target;" support a wide range of assurances to include a new "basic" level for low risk environments; and allow for a diverse approach to evaluations to provide for timely and cost-effective evaluations.

While, ITSEC, TCPEC, and the new Federal Criteria addresses individual national and regional interests, care has also been given in building on previous and existing criteria efforts. As a result there exists great potential for harmonization that can lead to a common basis for allowing for mutual recognition of product evaluations among participating nations.

PANEL: Perspectives on MLS System Solution Acquisition - A Debate by the Critical Players Involved

Joel E. Sachs
Arca Systems, Inc.
2841 Junction Ave., Suite 201
San Jose, CA 95134
408-434-6633

Panel Overview

Both the availability of MLS products and attempts at acquiring MLS system solutions have increased in recent years. Several of these acquisitions have already been deemed less than successful. A number of reasons have been suggested: integration of these products is not straight forward, defining and mapping mission requirements to security and system solution requirements is difficult, and certification and accreditation is hard and not uniform. Acquiring an MLS system solution that results in an accreditable secure solution is not simple; moreover, there is debate and confusion as to what should be specified during the initial phases of an acquisition that will help all parties involved throughout the life of the program. This panel will explore issues associated with developing a specification, statement of work, and evaluation criteria for successfully acquiring an MLS System Solution. The critical deliverables and their role in certification and accreditation will also be examined.

The panel will explore these issues by role-playing the critical players in the acquisition process, as opinions vary depending on one's position within the process. Each of the seven panelists will act on the behalf of an identified role with which they are experienced. These roles are: End-User Organization, Program Management Office, Advising Security Agency / Certification Body, Designated Approving Authority, Systems Integrator, Security Engineering Subcontractor, Vendor. The panel will discuss the issues associated with the pre-draft Request for Proposal [RFP], pre-RFP, pre-award, and post-award phases of an MLS System Solution acquisition. The panelists will discuss and debate their needs and concerns regarding the development of a MLS System Solution, with respect to the role that they are playing. Specific questions will be asked of the panel relative to each procurement phase.

Information is provided in the following sections to aid the audience with an understanding of the topics and issues of specifying, procuring, and accrediting MLS System Solutions. These sections include descriptions, example issues, and concerns of the critical players, as well as example critical questions for the panel.

Panel Roles, Descriptions, and Areas of Concern

End-User Organization

The end user organization has a requirement for a system solution. The results of this procurement will be delivered to this organization for their use.

Their main concerns are how to ensure that they get what they want, that it will be accreditable, how much will it cost, and how long will it take? They usually understand functional requirements reasonably well but often do not understand security and assurance requirements and security issues.

Program Manager's Office [PMO]

The PMO is the acquisition agency responsible for writing the RFP, awarding the contract, and supervising its execution. (Typically, a separate organization might be used to develop a system specification for the Statement of Work [SOW]. For the purposes of this panel, the player developing the specification will be considered merged with the PMO.)

The PMO's main concerns are system specification, cost, schedule, accreditation, and measuring the prime contractor's progress and compliance. The PMO understands the functional requirements as communicated by the end-users, but may not fully understand the security requirements, issues, and assurance needs that result from the mission and threat context.

Advising Security Agency / Certification Body

The Advising Security Agency is the End-User's and/or PMO's security arm that helps monitor the progress of the program to ensure that security within the program is adequately addressed. The Certification Body gathers the assurance evidence and performs risk analyses on the system. (For the purposes of this panel, these two roles have been combined as often happens in practice.)

Their main concern is whether the delivered system meets the security requirements specified in the RFP, security functionality and assurance. The certification body must provide enough evidence to allow the DAA to make a proper decision regarding its accreditation.

Designated Approving Authority [DAA]

The DAA is the individual responsible for the operational aspects of the system. It is this individual's responsibility to approve the system for operation.

The DAA's main concern is whether the system meets its operational requirements and its operational risk has been reduced to an acceptable level. Based on the evidence provided during the certification process, the DAA must make a decision whether the operational risk is acceptable given the evidence provided and the system's mission, and accredit or fail the system for operation. The DAA's accreditation of the system is his indication that he feels the risk is low enough or the operational need is high enough to allow the system to operate.

Systems Integrator

The Systems Integrator is responsible for the development and integration of the end-system as well as the management of all the subcontractors involved in the effort.

Their main concerns are how to provide the required functionality, security, and assurance within the budgetary and time constraints stipulated in the integrator's proposal. Other areas of concern include how to manage the security engineering effort to produce a functional and usable system and how to handle requested changes to the end-system.

Security Engineering Group/Subcontractor

Security Engineering is responsible for the security portion of the overall system development. This team is composed of internal systems integrator personnel, a security subcontractor, or a combination of both.

This team's main concerns are: how to relate component policies to the overall system policy, the trust requirements for each component, how to integrate trusted and untrusted systems, how to integrate multiple products into a single secure solution, and how to provide required assurance evidence. They may also be involved in determining the security requirements and policy, determining the appropriate assurance level, and how to provide assurance evidence.

Vendor

Vendors provide products that are used as part of end-user system solutions.

Their main issues are: how to relate their product features to the desired functionality and assurances needed within an MLS system solution and how to advise the systems integrator on the best use of these features.

Example Questions for Panel

Pre-Draft RFP Questions:

1) How should security, mission, and functional requirements and their interrelationships be stated and distinguished?

2) Should SOW state detailed security requirements, e.g., explicitly require segregation by either compartments or DAC, or should the SOW just simply state need to segregate planning from operations data?

3) How should the SOW handle the migration of data (i.e, the downgrading / transmission issue)?

4) Should trusted applications be explicitly required?

5) What can be done at this stage to ease the certification / accreditation process? Who should do it? How should it be requested?

6) How should threats be determined and documented? What information about threats should be provided to prospective bidders?

7) Who should identify or develop the following:

- Assurance Requirements
- Assurance Deliverable Schedule
- Security Architecture
- MLS Concept of Operations
- System-Wide Security Policy
- System-Wide Security Policy Model
- Certification and Accreditation Plan
- System Threat List and Risk Analysis

Who provides inputs, who writes, who reviews, who is the intended audience? When should these be done? Should the SOW be explicit? What should the DIDs require?

Considerations: a) It's more work for either the Specifier, PMO, Certification Body, or Systems Integrator; b) Not everything is known up front; c) If not done up front, bidders get to decide what is required, and some may use this flexibility to undercut other bidders by potentially deriving insufficient requirements.

Pre-RFP Questions:

8) Who should develop/determine the MLS Concept of Operations? The PMO, Advising Security Agency, or Systems Integrator? When?

9) What steps can be taken to ensure that an MLS system solution is proposed, not just an MLS operating system?

10) When should the Advising Security Agency, Certification Body, or DAA become involved? How and to what degree? At different stages who are they helping and to whom are they responsible? Should this be reflected in the RFP and SOW? How?

11) How and when should the overall assurance requirements be given? How should they be determined?

12) Should a Certification and Accreditation Plan be included in the RFP? If not, when should it be developed? How should it be specified that the system must be certifiable or accreditable?

Pre-Award Questions:

13) Should certification and accreditation be addressed in the proposal? How?

14) How should EPL (evaluated products list) ratings and status be handled? Should inclusion on the EPL be a strict requirement?

15) Which factors should be considered in the proposal evaluation criteria?

    a) the Technical approach? methodologies? architectures? trade-offs?

    b) the Assurance / Certification and Accreditation approach?

    c) the Participating Personnel?

16) As engineering process capability testing becomes routine, should security tests and exercises be administered as part of the evaluation of the bidders? If so, how should tests be given? If so, who should take the test? Should it be a group test?

Post-Award Questions:

17) How should the DAAs of the external systems to which the proposed system connects be dealt with?

18) How should the detailed security requirements be determined? How and when should they be delivered?

19) Where and how should security testing be integrated into the overall verification and validation of the system?

20) Should assurance testing be performed separately? When?

21) At what times within the development / certification process should assurance evidence be provided? Who is to review this evidence? How should it be developed?

22) How should component policies be related to an overall system policy?

23) How should assurance evidence be generated for an MLS System Solution that is composed of multiple trusted and untrusted products?

24) How can vendors provide functional capabilities to assist in the integration of their products into the system solution?

25) What assurance evidence can a vendor provide that enhances a product's appeal for use in a secure system solution for the System Integrator, Security Subcontractor, or Certifier / DAA?

Panel: Security Protocols for Open Systems

Chair: Paul A. Lambert, Motorola, Inc.

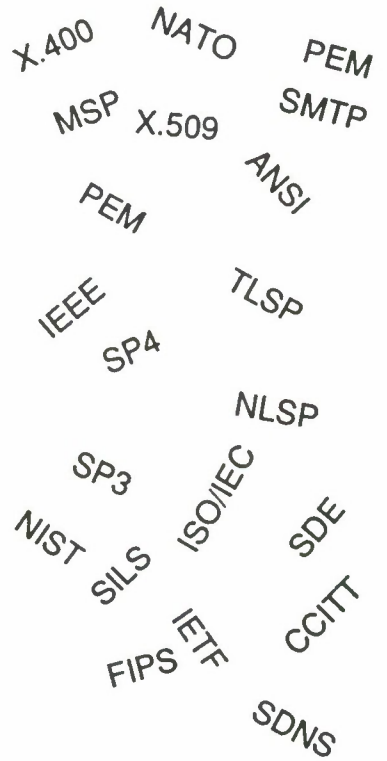
Panelists: David Solo, BBN,
Doug Maughan, NSA,
Russell Housley, Xerox,
Dale Walters, NIST,
Mike White, Booz Allen & Hamilton

"Open system security" may seem to be an oxymoron, but this contradiction in terms is being addressed by many standards organizations. Standards are being proposed for communication protocols that will cryptographically protect information. This panel will examine the application of these security protocols to protect communication systems.

One problem with selecting standards is that there are so many of them. Standards for security protocols are being pursued in groups that include ANSI, CCITT, IEEE, IETF, ISO/IEC, NATO, NIST, NIUF, and SDNS. The protocols that are being defined in these organizations include the Message Security Protocol (MSP), Network Layer Security Protocol (NLSP), Secure Data Exchange (SDE), Standard for Interoperable LAN Security (SILS), Security Protocol Layer 3 (SP3), Security Protocol Layer 4 (SP4), and the Transport Layer Security Protocol (TLSP).

In the Open System Interconnection (OSI) framework cryptography and security protocols are mechanisms to provide security services. These services include authentication, access control, confidentiality, integrity, and non-repudiation. Many of the security protocols provide almost identical security services. At least four "open" protocols are being developed to protect electronic messaging. Five more will provide end-to-end transmission security in the lower layers of the OSI reference model.

The panel will explore the tradeoffs in applying the various security protocols. The panel will also provide options and opinions for a security architecture, based on these protocols, for a variety of communication environments. All panelists are active contributors to their respective topic areas. It is hoped that their contributions will serve as a useful introduction to the application of security protocols.



Executive Summary

Panel
"TMach" A Symbol of International Harmonization

Ellen E. Flahavin, Chair, *NIST*

Panelists:

Brian Boesch, *DARPA*

Dr. Martha Branstad, *Trusted Information Systems, Inc.*

C. Ketley, *U.K. Government*

Klaus Keus, *German Government*

Trusted Mach (TMach) is a highly portable trusted operating system being developed by Trusted Information Systems, Inc. (TIS) under DARPA funding. In November 1990, officials from the national Institute of Standards and Technology (NIST), met with European officials and TIS to discuss initiating a preliminary developmental evaluation of TMach against the Information Technology Security Evaluation Criteria (ITSEC). As a result of this meeting, NIST finalized an agreement with the U.S. Defense Advanced Projects Research Agency (DARPA) to coordinate and oversee the TMach evaluation work to be done by Germany and the United Kingdom. Under the agreement, NIST receives funds from DARPA for the evaluations, negotiates the contracts with appropriate organizations in those countries, monitors the TMach evaluation contract performance for DARPA, and provides the contractual and management interface with the U.S. Government. In essence, NIST has become the contractual "middle-man" for DARPA. NIST is responsible to DARPA as the u.s. government agent for setting the evaluation tasks and coordinating the evaluation work.

German and United Kingdom evaluation contractors were selected and the ITSEC evaluations of TMACH began in September of 1991. these multiple evaluations have been undertaken with he objective of understanding the different criteria and evaluation processes by seeing how they relate to a single system.

TMach, which will provide users with both high-level trust and a Unix interface, is under evaluation by four different nations against three separate criteria. In the U.S., TMach is being evaluated by the B3 level against the Trusted Computer System Evaluation Criteria (TCSEC). Concurrent evaluations of TMach at F-B3/E5 against the Information Technology Security Evaluation Criteria (ITSEC) are on-going in the U.K. and Germany.

TMach is also being evaluated against the Canadian Trusted Computer Product Evaluation Criteria in Canada. Claims are made that the various criteria are compatible with one another. By actually evaluating TMach against the three criteria, the different requirements of each criterion and evaluation process become visible. If desired, efforts to reconcile the differences can then be undertaken based upon detailed knowledge and understanding.

The TCSEC and Canadian evaluations were initiated early in 1992. Experience thus far has uncovered several fundamental differences between the TCSEC and the ITSEC. The ITSEC imposes different documentation requirements, and distinctions in the evaluation process. These and other differences will be discussed in the panel presentation.

Executive Summary

The Trusted Product Evaluations Program Process Action Team

S. Nardone, *National Security Agency*

In March of 1992, at the direction of the NSA Deputy Director for Information Systems Security, Dr. James Hearn, a Total Quality Management (TQM) Process Action Team (PAT) embarked on a six-month effort to enhance the effectiveness of the National Security Agency/National Computer Security Center (NSA/NCSC) Trusted Product Evaluations Program (TPEP).

Since its inception in 1983, the TPEP has evaluated commercially available trusted products. These trusted products are Commercial-Off-The-Shelf (COTS) solutions providing security in the form of functionality and assurance for use in classified and/or sensitive system applications for the Department of Defense (DoD) and Intelligence Community (IC). The TPEP was established to help put trusted products into the hands of Automated Information System (AIS) users with risk. Its mission is to evaluate systems based on the Trusted Computer System Evaluation Criteria (TCSEC) and its Interpretations resulting in a level of trust rating (C1-A1) and placement on the Evaluated Products List (EPL). This list is posted on DOCKMASTER, the flagship information system of the NCSC, and distributed in a quarterly INFOSEC Product and Services Catalog published by NSA.

After ten years of performing trusted product evaluations, the NSA/NCSC has decided to reassess its ability to meet TPEP customer needs. The Process Action Team will be briefing the national computer security community on the results of this six-month effort as a part of the 15th National Computer Security Conference. This briefing will provide insight into the background, methodology and criteria used throughout the process by the PAT. In addition, the PAT will detail the findings and implementation plans for any and all recommendations.

Executive Summary

PANEL: Virus Attacks and Counterattacks Real-World Experiences

James P. Litchko, Chair
Director of Business Development
Trusted Information Systems, Inc.

Ms. Janet Keys
Computer Security Manager
Headquarter NASA

Ms. Louise Mandeville
Computer Systems Manager
Miller, Balis & O'Neil, P.C.

Mr. George Wellham
EDP Audit Manager
MNC Financial, Inc.

Over the past two decades, viruses have migrated from fiction to our working environment. With the open availability of virus tool kits, virus cookbooks, and creative programmers, experts believe that the number of unique viruses now circulating is over 2000 and suspect that new viruses are being developed at rate of two a day. Some project that there will be over 10,000 viruses by the year 2000. Corporate profits and credibility levels now have a direct relationship to the reliability and availability of information systems and the integrity of their data. With the trend to link the systems over networks, the losses resulting from a single virus attack increases from thousands of dollars to millions.

What is required is open discussion of experiences to increase the understanding of the threat and effective prevention and reactions. In the past, few would openly admit that they had been a victim of a virus attack, less would openly talk about the specifics of an attack. All of the articles and discussions about the expected losses and effectiveness of counter-measures are very interesting, but very impersonal and they do not truly answer the real questions:

- What really happens during and after an attack?
- How effective were those countermeasures?
- Where were the pit-falls?
- What is the real impact of a virus attack?
- Are there effective pre-attack legal countermeasures?
- What were the near-term and long-term effects?

Each of the panelists were victims and survivors of actual virus attacks on real-world computer and network systems in both commercial businesses and government agencies. During the panel session, each member will describe the attack on their systems and their actions to counter-attack. The corrective actions that will be discussed by the panel will include those that are technical, procedural, organizational, and legal. Through interactive discussions with the audience and the panelists, the panel will provide their prospectives and answers to the above questions and to additional audience questions.

*U.S. G.P.O.:1992-625-512:60546